Coupling problem in thermal systems simulations

TREBALL FINAL DE MÀSTER

Adrià González Esteve

Màster Universitari en Enginyeria Industrial, UPC

Co-directors: Xavier Trias i Assensi Oliva

Ponent: Enrique Velo

Convocatòria de Tardor

Gener 2017

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Abstract

Building energy simulation is playing a key role in building design in order to reduce the energy consumption and, consequently, the CO_2 emissions. An object-oriented tool called NEST is used to simulate all the phenomena that appear in a building. In the case of energy and momentum conservation and species transport, the current solver behaves well, but in the case of mass conservation it takes a lot of time to reach a solution. For this reason, in this work, instead of solving the continuity equations explicitly, an implicit method based on the Trust Region algorithm is proposed. Previously, a study of the properties of the model used by NEST-Building software has been done in order to simplify the requirements of the solver. For a building with only 9 rooms the new solver is a thousand times faster than the current method.

Index

1 Introduction

In 2014 the dominant end-users of energy in EU-28 were: transport (33,2%), industry (25,9%) and households (24,8%) (see Figure 1.1). This means that there is a huge potential of energy saving in domestic consumption.

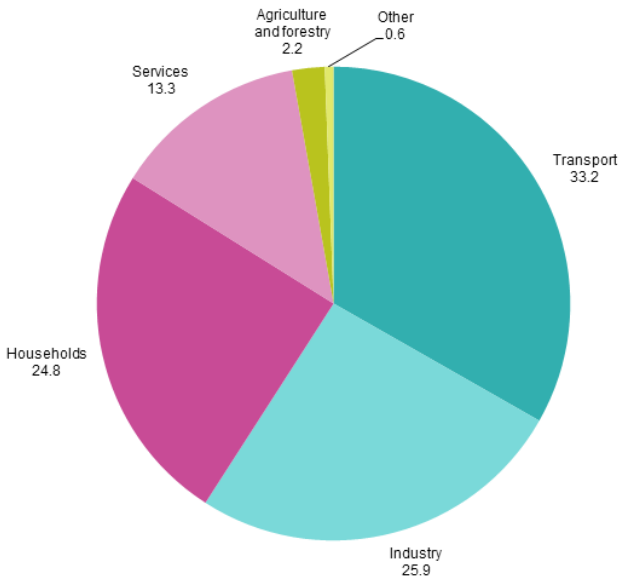


Figure 1.1: Final energy consumption, EU-28, 2014 (based on tonnes of oil equivalent). [5]

Almost the half of buildings energy consumption comes from Heating Ventilation and Air Conditioning (HVAC) systems [18]. So building energy simulations are important to optimize the design of HVAC systems and its control, that results in a reduction of the energy demands [19]. According to Capdevila *et al.* [3], numerical simulations "can give vital information of the peak loads during the heating and cooling seasons, room temperatures and velocity distributions for maintaining an adequate indoor environment, and overall energy demands during a year".

Damle in his thesis [4], defines the building energy simulation as "the science of estimating the energy interactions within a building". These interactions can be with the surroundings due to the changing weather, with the occupants due to their activities and with the HVAC systems that try to counteract the thermal loads. Furthermore, because of its design and construction, the buildings have some inertia, so the problem involves a dynamical system. To solve it, the interactions must be modeled and the complexity of these models depends on the requirements of the project. For example, a room can be treated as a 0D problem where pressure and temperature are considered homogeneous or as a 3D problem where a Computational Fluid Dynamics (CFD) software is used for detailed computations. The first case is a simple model with less degrees of freedom and its computation is fast, but maybe, there are some phenomena that are not modeled accurately such as turbulence or convection. So there is always a dispute between computational cost and accuracy.

To achieve a reasonable balance between them, some software use a modular object-oriented simulation tool [3, 4, 10, 19]. This tool lets reduce the whole building model into simpler sub-models linked with each other. So the problem consists of a network of "objects" that solve every sub-model and, to couple them [3], the boundary conditions are exchanged with the neighbour elements. As López said in his thesis [10], "this partitioned approach provides an efficient and practical way to simulate buildings. Unlike monolithic approaches, where all phenomena are resolved by means of a single numerical code, this strategy is much flexible since it allows rapid setup of new buildings configurations". Moreover, the elements can be modeled with different levels of detail and the code can be parallelized [19]. In Figure 1.2 there is a scheme of the idea of treating a room as a network of objects.

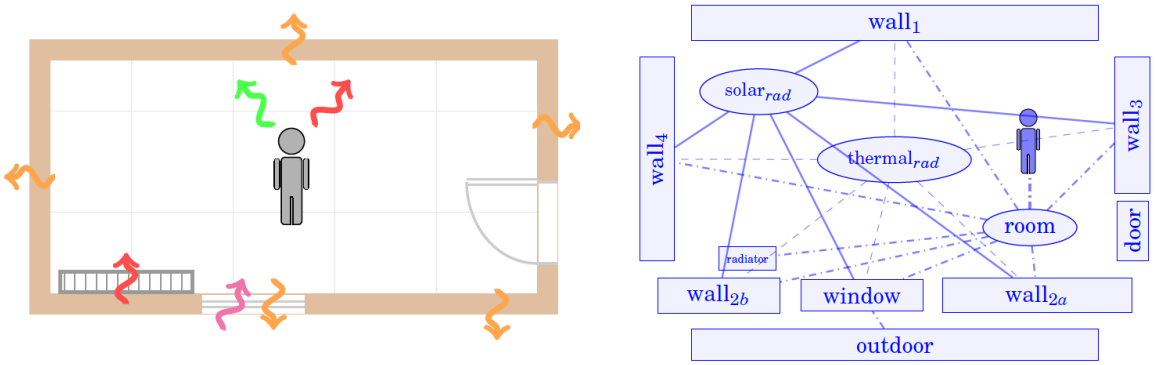


Figure 1.2: Scheme of a room and its representation in a modular object-oriented tool. [10]

At the Heat and Mass Transfer Technological Center (CTTC), an object-oriented platform called NEST (Nest Element Specialization Toolkit) has been designed to solve multiphysics problems. Its application in building simulations works well in solving the governing equations of momentum conservation, energy conservation and species transport. But this modular approach applied to mass conservation has problems of computational cost. For this reason, it has been decided to make an exception and solve the mass conservation equations monolithically, instead of using a modular approach. Then, in this thesis, a study of the governing equations of continuity has been carried out, in order to choose the most appropriate solver.

First, in Chapter 2, the NEST-Building structure is explained, where the current solver used by NEST is described in Section 2.3. Because of the limitations of this method, in Chapter 3, some non-linear solvers have been studied to ensure that the chosen one complies with the restrictions of accuracy. Finally, in Chapter 4 it is shown that the proposed methods are really faster than the current solver used by NEST-Building. In fact, there is one method faster than the others.

2 NEST-Building model

In this Chapter is going to be introduced the mathematical model used in NEST-Building for natural ventilation. NEST-Building sub-models are in charge of solving the conservation laws of mass, momentum and energy. There are sub-models for walls, for example, used in the energy balance that "solve the thermal interaction between the rooms and between the rooms and the exterior" [10]. Despite this, in this work, as the goal is to improve the conservation of mass, only the NEST-Building sub-models involved in mass conservation are considered and, consequently, explained. Although focusing only in the *Rooms* and *Openings* sub-models, NEST-Building has other sub-models that solve the equations of momentum and energy that are not treated in this work.

First, the NEST structure is going to be explained to see how the large problem is split in smaller sub-problems and then, the mathematical models that explain the air flow in a building are going to be deduced. Finally, the current solver used in NEST-Building for mass conservation is going to be described. The NEST structure is based on the theory used in Finite Volume meshes [12], the air flow models use the theory of flow through an orifice [16] (that is adapted to a building situation) and the NEST-Building solver uses the SIMPLE method to ensure mass conservation.

2.1 NEST-Building structure

NEST is an object-oriented tool that "consists on a set of individual numerical sub-models that solve the conservation laws of mass $[\cdots]$ along the buildings spaces and components" [10]. As NEST is a general toolkit, instead of treating them like buildings spaces or components, they are called objects or elements. Specifically, the sub-models that NEST-Building uses to model the natural convection through a building are the *Rooms* and *Openings* objects.

A *Rooms* object is a sub-model applied to a single room, where the conservation laws are solved in a control volume that contains air at uniform conditions of pressure and temperature. To conserve the mass in a room, this sub-model needs to know the air flow through their doors and windows. Doors and windows use the same *Openings* sub-model to compute the air flow, because in fact, both of them link two zones.

An *Openings* sub-model determines the air flow between two rooms (a door) or the air flow between a room and the outside of the building (a window). This means that the building, if only natural convection is considered, is divided in two types of sub-models that are connected depending on the distribution of the building. In Figure 2.1 there is an example of the connectivity of these elements in a house.

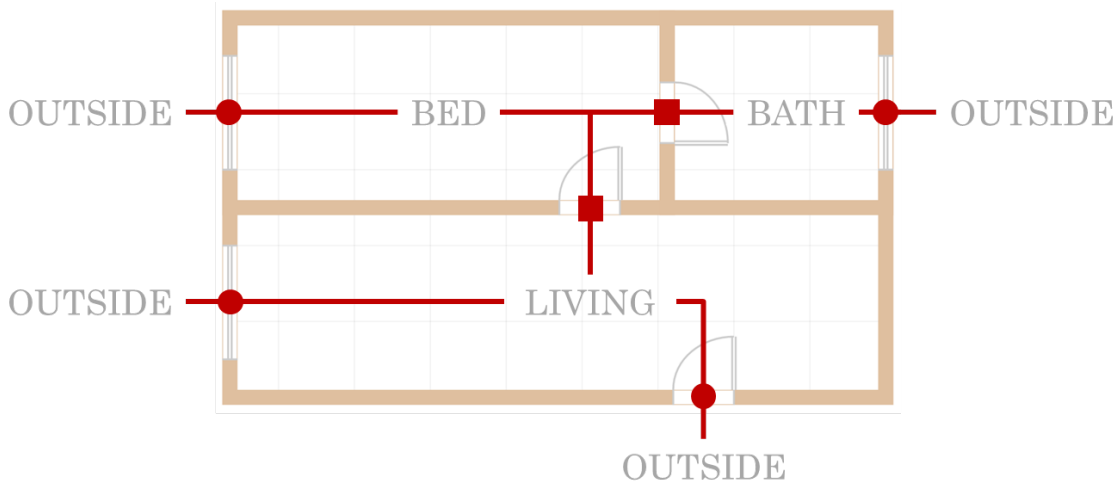


Figure 2.1: Example of NEST-Building connectivity. Based on [10].

It is observed that there are two kind of *Openings*: windows (represented by circles) that connect a *Rooms* object with the exterior of the building and doors (represented by squares) that connect two interior *Rooms*. In Figure 2.1 the living room has physically an access door that connects it with the exterior of the building but, due to the topology of NEST, as it connects a room with the exterior, NEST considers it as a window. That is why this door is represented by a circle instead of a square. This distinction between doors and windows is important during the definition of the NEST-Building geometry. If *Rooms* are considered as control volumes and *Openings* as boundary conditions, there is a parallelism with Finite Volume Method because every *Rooms* object corresponds to a very large element and the *Openings* correspond to the faces of the Finite Volume mesh. This parallelism is illustrated in Figure 2.2.

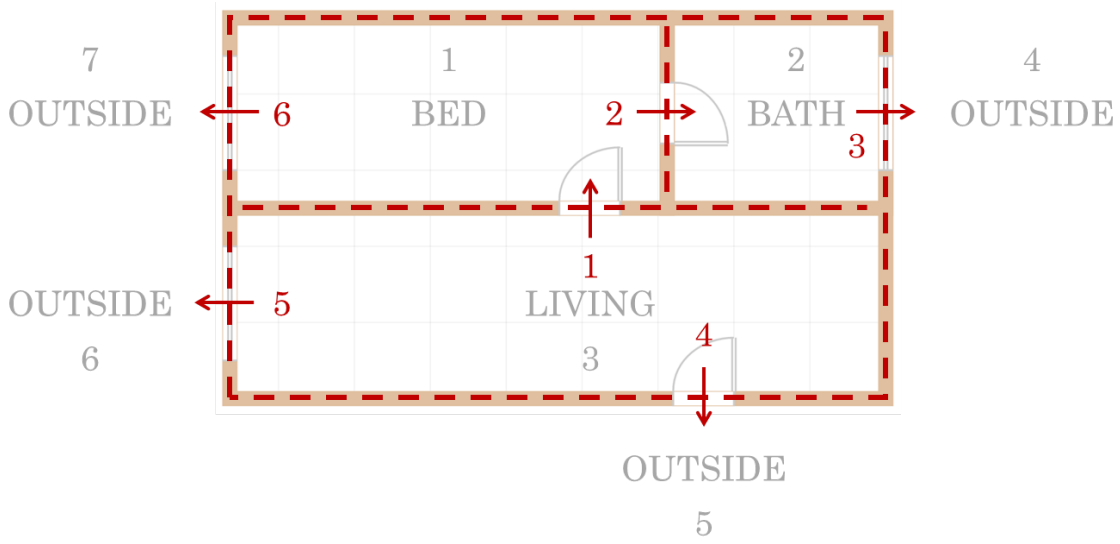


Figure 2.2: Example of a NEST-Building mesh. Based on [10].

Then, interior boundaries (a door) and exterior boundaries (a window) must be distinguished to use the same topological structure as in a Finite Volume mesh. *Openings*, as faces in a Finite Volume mesh, must be oriented when they are defined. The orientation of an *Openings* object is such that the arrow points from one *Rooms* object called owner to a second *Rooms* object called neighbour [12]. For example, in Figure 2.2, door 1 connects the living room (owner) with the bed room (neighbour). Otherwise, windows have always as owner a *Rooms* object and the outside as a neighbour (all the arrows point outwards). In the case of *Rooms*, every *Rooms* object has a set of neighbours with which there is an exchange of air through an *Openings* object. For example, the bathroom is connected with the bedroom via door 2 and with the outside 4 via window 3. It is necessary to mention that the numeration, for both *Rooms* and *Openings*, begins with the interior objects and finishes with the objects related with the outside.

In the case of windows, there is a bijection between *Openings* and outside *Rooms*. For this reason, they are sorted in the same way. In other words, the first window (*Openings* object 3) has as neighbour the first outside *Rooms* object and the last window (*Openings* object 6) has as neighbour the last outside *Rooms* object.

2.2 NEST-Building sub-models

As said in the previous Section 2.1, NEST-Building divides a house in *Rooms* and *Openings*, in order to reduce the monolothical problem to a simpler modular approach. In *Rooms* the mass conservation law is solved, while the mass flow rate through contiguous *Rooms* is computed by the *Openings*. They are responsible for coupling the problem, exchanging boundary conditions between *Rooms*. Figure 2.3 shows an example of a very simple multizone building.

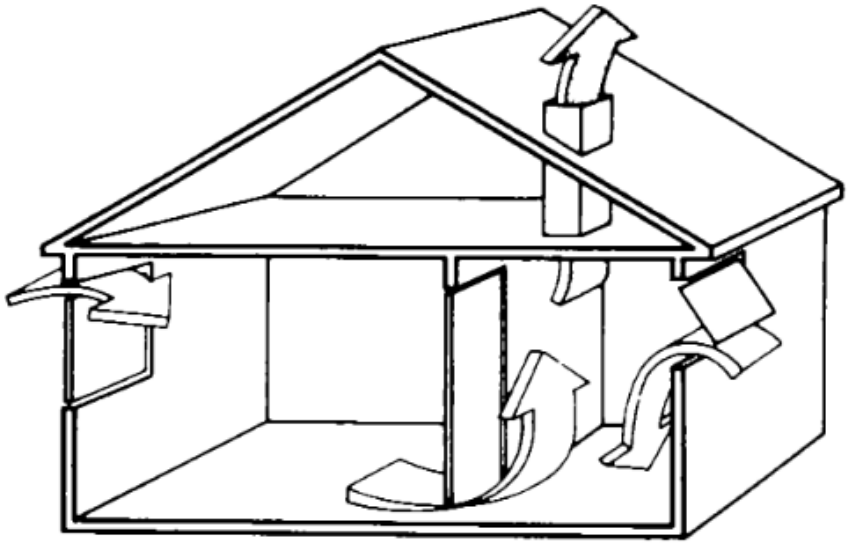


Figure 2.3: Example of air flow in a building [1].

2.2.1 NEST *Rooms* sub-model

Every *Rooms* object sub-model is in charge of solving the conservation of mass in a given room. The fact of considering the control volume with a unique temperature and pressure, reduces the *Rooms* object to a lumped model that depends on the interaction with the neighbour *Rooms* (in Figure 2.4 there is a scheme of mass conservation). Despite being a simple model, it gives good results and that it is why it is widely used [1, 3, 4, 6, 9, 10].

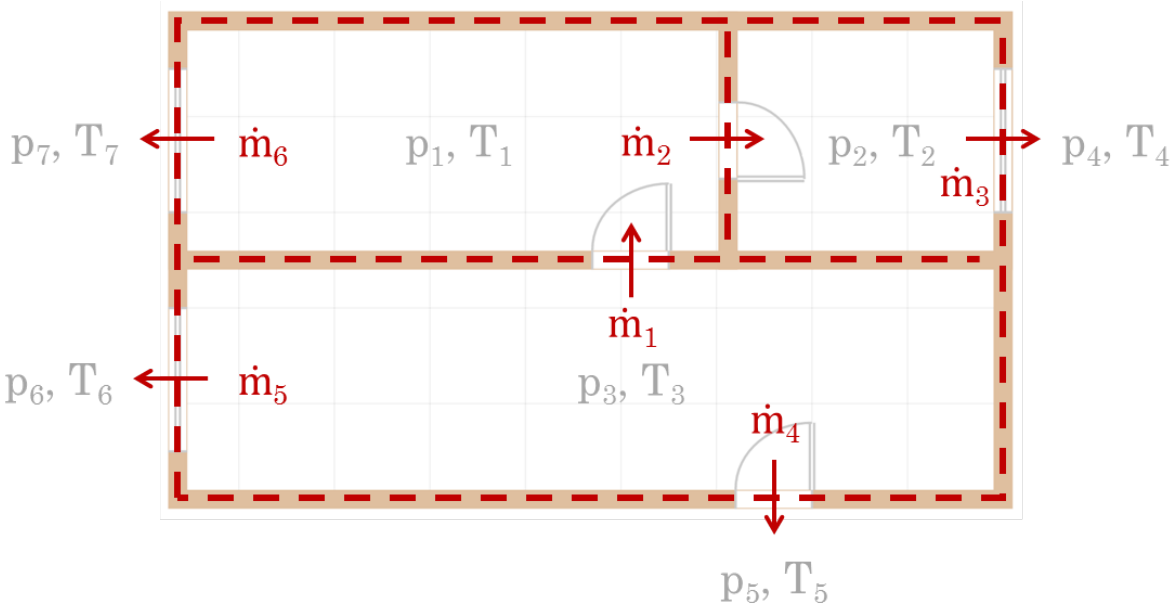


Figure 2.4: Example of mass conservation in a building [1].

The rooms are considered at constant pressure and temperature because, as will be discussed later in Sub-Section 2.2.2, the model used for the air flow through the *Openings* only depends on these variables. Turning to the *Rooms* sub-model, as the *Openings* objects are placed at the boundaries of the control volume, the equation of continuity is [3, 10]

$$\frac{\partial m_P}{\partial t} = \sum_{k \in Inlet} \dot{m}_k - \sum_{k \in Outlet} \dot{m}_k, \quad \forall P \in Rooms\,. \tag{2.1}$$

In other words, the accumulation of air in a room P is equivalent to the mass flow exchanged with the neighbour *Rooms*. The contribution of the mass flow rate depends on whether the opening is an outlet or an inlet and this, in turn, depends on which is the upstream room, the owner or the neighbour.

This accumulation of air in the room entails a change of the density because the volume of a room remains constant. Nevertheless, in next Chapter 4, they will be applied to situations similar to the ones in [1, 6, 9]. In all these cases, the *Rooms* are considered at constant temperature. So, in this work, temperature is considered as a given parameter. Then, as density depends on

temperature (2.10), the problem (2.1) becomes stationary.

$$\frac{\partial m_P}{\partial t} = \frac{\partial \rho(T_P)}{\partial t} V_P = \sum_{k \in Inlet} \dot{m}_k - \sum_{k \in Outlet} \dot{m}_k = 0, \quad \forall P \in Rooms\,. \tag{2.2}$$

As it is observed, the *Rooms* sub-model depends on the air flow through the *Openings* objects.

2.2.2 NEST *Openings* sub-model

According to Allard *et al.* in [1], "air flow through the rooms of a building stems from pressure distribution around and within the building itself". Specifically, air circulation is driven by pressure gradients going from zones of high pressure towards zones of low pressure.

To find the relation between the flow rate and the pressure gradient, the doors and windows are going to be considered as a restriction of the air flow through the house. Then, the theory of flow through orifices in pipelines [16] can be a good approximation to model the mass flow rate through an opening in the building (see Figure 2.5).

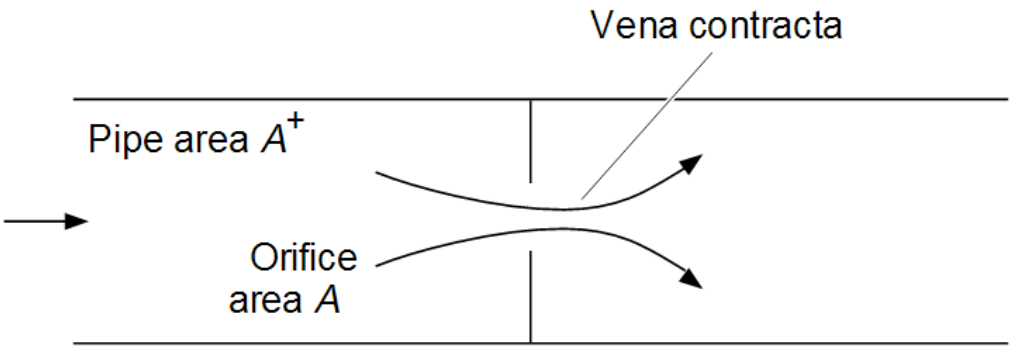


Figure 2.5: Flow through an orifice [16].

The theory of orifices is based on the Bernoulli's equation. Taking a streamline that crosses the opening, if the density is taken from the upstream room, the Bernoulli's equation is

$$p^+ + \frac{1}{2}\rho^+ v^{+2} = p^- + \frac{1}{2}\rho^+ v^{-2}, \tag{2.3}$$

where superscript + indicates that the variables belong to the upstream room and - to the downstream room. In equation (2.3) the variable p corresponds to the stagnation pressure [6, 9] and its expression will be discussed later.

In the Bernoulli's equation (2.3), the velocities v^+ and v^- are related because mass is conserved through the opening

$$Q = A^+ v^+ = A\, v^-\,, \tag{2.4}$$

where A^+ is the cross-section area of the upstream room and A is the cross-section area of the

opening. Then, replacing (2.4) into (2.3), the pressure drop through the opening is

$$p^+ - p^- = \frac{1}{2}\rho^+ \left(\frac{Q}{A}\right)^2 - \frac{1}{2}\rho^+ \left(\frac{Q}{A^+}\right)^2 . \tag{2.5}$$

Now, if the flow rate is isolated, it can be observed that the air flow is driven by the difference of pressure between the contiguous rooms

$$Q = A\sqrt{\frac{1}{1-(A/A^+)^2}}\sqrt{\frac{2\,(p^+ - p^-)}{\rho^+}} . \tag{2.6}$$

In order to get an expression more general independent of A^+ and to take into account the discrepancy between theory and reality, the coefficient of discharge C_d is introduced. The coefficient of discharge C_d is a constant that simplifies the expression of the mass flow rate through an opening [9]

$$\dot{m} = \rho\, Q = C_d\, A\sqrt{2\,\rho^+\,(p^+ - p^-)} . \tag{2.7}$$

where C_d varies from 0,4 to 0,6 depending on the difference of temperature between rooms. This range of C_d was observed by Wilson and Kiel in their experiments [7]. Otherwise, in this work, as the main objective is to try different non-linear solvers, the coefficient of discharge is going to be fixed to $C_d = 0{,}5$.

As said in the previous Sub-Section 2.2.1, the computation of the mass flow rate between two rooms (2.7) depends on the the pressure of the neigbhour room. According to Allard *et al.* [1], "pressure distribution is due to the combined actions of wind, thermal buoyancy (stack effect) and mechanical ventilation, if present". In this work, as there is natural ventilation, only the effects of wind and temperature gradients are going to be considered [3]. In the case of wind, due to its velocity, it can be observed Figure 2.6 that it draws a pressure profile around the buildings.

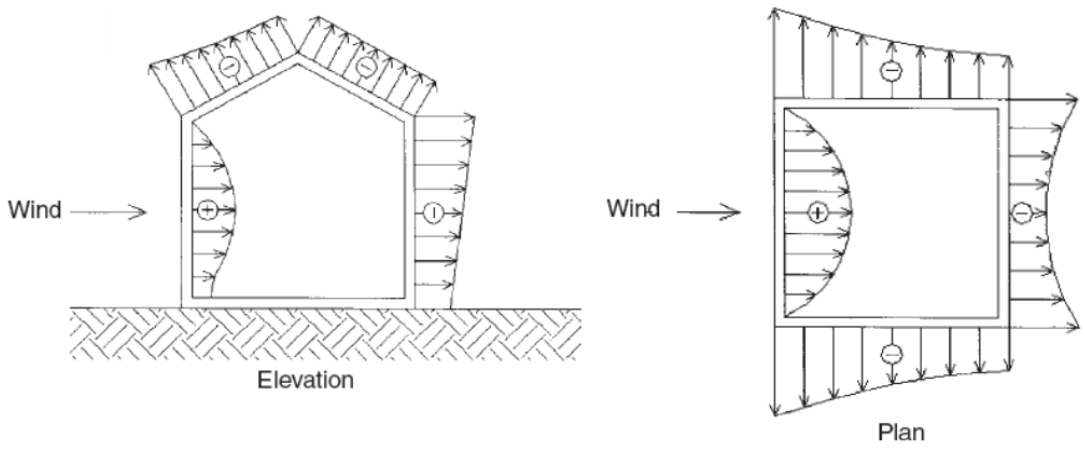


Figure 2.6: Pressure distribution around a building [9].

This is because the pressure that appears in equation (2.7) is referred to the stagnation pressure, that corresponds to the sum of static and dynamic pressure

$$p = p_{\text{static}} + \frac{1}{2}\rho v_{\text{wind}}^2 \,. \tag{2.8}$$

Using p reduces the number of variables because instead of considering the p_{static} and p_{dynamic} as two independent variables, the effect of wind is introduced in the stagnation having only one variable. Then, if the velocity outside the building is known and the outside of the house is considered at atmospheric pressure, the stagnation pressure can be imposed as a boundary condition.

Inside the building, the velocity of the air is not so high and the pressure is driven by the differences of temperature in the room. The relation between pressure and temperature is obtained using the Bernoulli's equation in a room

$$p = p_0 - \rho(T)\, g\, h, \tag{2.9}$$

the 0 subscript is used to highlight that it corresponds to the stagnation pressure at the reference level where the hydrostatic pressure is zero. So, if buoyancy is to be taken into account, the stagnation pressure at the reference level p_0 should be the variable of the problem, instead of pressure p.

In the expression (2.9), the effect of wind is included using p_0, while the second term, is the one that corresponds to the thermal buoyancy [2] because the density depends on temperature (h, a given parameter, corresponds to the height of the opening with respect to the reference level). For simplicity, in this work is going to be supposed that air behaves as an ideal gas, so density the product $\rho \cdot T$ is constant

$$\rho(T) = \rho(T_0)\frac{T_0}{T}. \tag{2.10}$$

In this case, the 0 subscript is used to indicate that T_0 is a reference temperature. The reference temperature is 20 °C and it corresponds to an air density of 1.204 kg m^{-3}. This variation of density does not conflict with the assumption of incompressibility taken in the Bernoulli's equation because all the streamlines are at constant temperature. The streamline that crosses the opening (2.3) is considered at the upwind temperature, while the streamlines corresponding to (2.9) have the temperature of the room to which they belong.

Then, replacing (2.9) and (2.10) in (2.7), a relation between the mass flow rate and the pressure distribution is obtained

$$\dot{m}\left(p_0^+, p_0^-\right) = C_d\, A \sqrt{2\,\rho(T_0)\frac{T_0}{T^+}\left(p_0^+ - p_0^- - \rho\left(T_0\right)\, T_0\left(\frac{1}{T^+} - \frac{1}{T^-}\right) g\, h\right)}, \tag{2.11}$$

where the temperature of the rooms, the size of the opening and its height are treated as given parameters. So the flow rate between them only depends on their pressure and always blows from high to low pressure. It should be underlined that the expression (2.11) depends on which is the upstream/downstream but in the NEST "mesh", the *Openings* have neigbhour/owner rooms. Then, if the neigbour room corresponds to the upstream, the mass flow rate is the opposite.

From now on, for simplicity, instead of using the variable p_0, it will be recovered the pressure p because their difference is a constant, the buoyancy term $\rho\left(T\right)\, g\, h$, and the partial derivatives coincide $\dfrac{\partial}{\partial p_0} = \dfrac{\partial}{\partial p}$. Then, the expression (2.7) will be used instead of (2.11), but in the moment of replacing p, the expression (2.9) will be used. The model (2.7) is a first approximation of the mass flow rate, but there are some studies [1, 2, 6, 9] that use a more general expression

$$\dot{m}\left(p^+, p^-\right) = K\left(p^+ - p^-\right)^n, \tag{2.12}$$

where K is a constant and the exponent n goes from 0,5 to 1 [6] and is usually fixed to $n = 0,65$ [6, 9] instead of the $n = 0,5$ taken in the first approximation (2.7). Otherwise, (2.7) and (2.12) share properties of symmetry with respect to the variables p^+ and p^-. Furthermore, their derivatives are computed in the same way (power rule), so the first approximation model (2.7) is enough to try different solvers because, as discussed later, the different solvers depend on the partial derivatives.

2.3 NEST-Building solver

Once the model used by NEST-Building is introduced, in this Section, the current solver of NEST-Building is going to be described and the disadvantages of this method are going to be discussed. NEST uses the SIMPLE method [15] to couple the momentum and the mass conservation laws. Applying this method to the benchmark problem [1, 6, 9] does not make any sense because there is no momentum equation. Otherwise, SIMPLE method is going to be described to see the weaknesses of this method.

2.3.1 SIMPLE algorithm

The SIMPLE algorithm is an iterative method that in every step corrects pressure and mass fluxes

$$p_P = p_P^\star + p'_p, \tag{2.13a}$$

$$\dot{m}_k = \dot{m}_k^\star + \dot{m}'_k, \tag{2.13b}$$

the $\star$ superscript corresponds to the guessed values and the apostrophe is used to identify the correction of the pressure p_P and the mass flux $\dot{m}_k$. To reduce the number of variables, the correction of the mass flow is going to be related with the correction of the pressure.

As seen above in Sub-Section 2.2.2, the mass flow rates through the openings $\dot{m}_k$ depends on the pressure of the contiguous rooms (2.11). Then, using the chain rule, the variation of mass flow rate depends on the variation of the pressures. Otherwise, as NEST is modular and the *Rooms* sub-models are solved in parallel, the correction of mass only takes into account the correction of the current room P

$$\dot{m}_k = \dot{m}_k^\star + \frac{\partial \dot{m}_k^\star}{\partial p_P^\star} p'_P. \tag{2.14}$$

As both corrections (2.13a) and (2.14) only depend on the correction of the pressure, the continuity equation over the room P depends on the correction of the pressure

$$\sum_{k \in Inlet} \dot{m}_k^\star - \sum_{k \in Outlet} \dot{m}_k^\star + \sum_{k \in Inlet} \frac{\partial \dot{m}_k^\star}{\partial p_P^\star} p'_P - \sum_{k \in Outlet} \frac{\partial \dot{m}_k^\star}{\partial p_P^\star} p'_P = 0. \tag{2.15}$$

Then, as the correction of the pressure is a common factor, it can be isolated easily

$$p'_P = -\frac{\sum_{k \in Inlet} \dot{m}_k^\star - \sum_{k \in Outlet} \dot{m}_k^\star}{\sum_{k \in Inlet} \frac{\partial \dot{m}_k^\star}{\partial p_P^\star} - \sum_{k \in Outlet} \frac{\partial \dot{m}_k^\star}{\partial p_P^\star}}. \tag{2.16}$$

As said before in Sub-Section 2.2.2, this algorithm depends on the partial derivative of mass flux with respect to the pressure. So there is not much difference between taking (2.7) or (2.12) for the mass flow rate, because their derivatives follow the power rule. In practice, their performance should not be different.

Once the pressure correction is obtained, the mass flux rate and the pressure are corrected iteratively using the equations (2.13a) and (2.14), until the correction of the pressure (2.16) is less than a prescribed tolerance. In Figure 2.7, to explain it better, there is a flow chart of the SIMPLE algorithm used by NEST-Building.

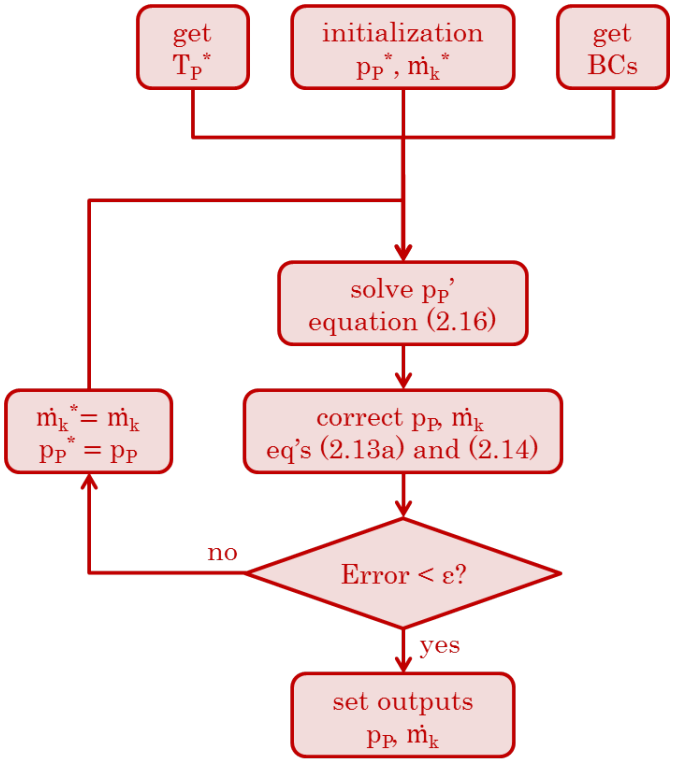


Figure 2.7: Building-NEST SIMPLE algorithm.

2.3.2 Disadvantages of the current solver

The main problem of considering the SIMPLE algorithm is that the current modular approach does not take into account the change rate of the neighbour rooms. This slows downs the algorithm because more iterations are needed to get convergence.

But the biggest disadvantage is the propagation of the information. Usually, if a large cloud moves in front of the sun, there are some rooms that begin to cool and this effect is propagated to the other rooms of the building progressively, first the neighbour rooms and then, the others. Otherwise, if a window is opened, a heavy current of air appears and there is a door that ends up slamming on the other side of the house. This is because the air is incompressible and, due to mass conservation, this change of mass flow is automatically transmitted to all the house.

This means that the SIMPLE algorithm is not ideal to solve the mass conservation because the communication through the *Rooms* should be automatic. This inconvenient does not appear in momentum and energy conservation, because in a house there is more inertia in these phenomena. It is widely known that warming a house during the winter is not instantaneous. Then, in order to get a solution more quickly but with the same accuracy, in this work, a non-linear solver is proposed. Instead of solving the mass conservation using a modular approach, the continuity law is going to be solved monolithically.

3 Non-linear solver

In order to spread the information through the *Rooms* instantly, an implicit method is proposed. The problem (2.2) consists of a system of non-linear equations because of the non-linearity of the mass flow rate (2.11). Every equation of the system corresponds to the mass conservation in a *Rooms* object and the variables of the problem are reduced to the pressure of the *Rooms* because the mass flow rates depend on the gradients of pressure. The problem is totally coupled and it reduces to find the vector $\vec{p}$ that conserves the mass in all the *Rooms*, $\vec{F}(\vec{p}) = 0$. In order to know the state of the art of non-linear solvers, the methods used by MATLAB have been studied [11]. Finally, the chosen algorithm will be described in detail.

3.1 State of the art

Non-linear solvers attempt to solve the system of equations by minimizing the sum of squares of the components [11]

$$\min_{\vec{p}\in\mathbb{R}^n} f(\vec{p}) = \frac{1}{2}\left\|\vec{F}(\vec{p})\right\|^2. \tag{3.1}$$

This reduces to an optimization problem because both problems are equivalent. If the sum of squares is zero, the system of equation is solved [11].

For further calculations, the gradient of f is

$$\frac{\partial f}{\partial p_i} = \frac{1}{2}\frac{\partial}{\partial p_i}\left[\sum_j F_j^2\right] = \sum_j F_j \frac{\partial F_j}{\partial x_i} = \sum_j J_{ji}F_j \Rightarrow \vec{\nabla} f = J^T(\vec{p})\,\vec{F}. \tag{3.2}$$

There are two common strategies: line search and trust region methods. In the line search method, "the algorithm chooses a direction d_k and searches along this direction from the current iterate p_k for a new iterate with a lower function value" [13]. In this case, the minimization problem is

$$\min_{\alpha_k>0} f\left(\vec{p}_k + \alpha_k \vec{d}_k\right). \tag{3.3}$$

Given a trial direction $\vec{d}_k$, the line search method becomes a one-dimensional problem that consists of finding the distance to move α_k [13]. The problem of this method is that, maybe, the chosen $\vec{d}_k$ is not a downhill direction so the step will not improve the solution.

On the other hand, in Trust Regions methods, the minimization search is restricted to some neighbourhood around the current iterate p_k. This lets try more than one direction. Then, the minimization problem is

$$\min_{\left\|\vec{d}_k\right\|<\Delta_k} f\left(\vec{p}_k + \vec{d}_k\right). \tag{3.4}$$

If the solution is updated, the trust region is expanded; but if there is no improvement of the solution, the trust region is shrunk and the problem is re-solved. So the Trust Region method automatically adapts to the performance of the algorithm.

Both methods are somehow dual because line search methods choose a direction "and then focus their efforts on finding a suitable step length α_k along this direction", while Trust Region methods "define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this trust region" [13].

3.1.1 Steepest Descent method

The most intuitive line search method is the one that takes the Steepest Descent direction, because is the direction along which f decreases most rapidly [13]. If the Taylor approximation of f is considered (with only two terms)

$$f\left(\vec{p}_k + \alpha_k \vec{d}_k\right) \simeq f\left(\vec{p}_k\right) + \alpha_k \vec{\nabla}^T f_k \, \vec{d}_k, \tag{3.5}$$

the rate of change of $f\left(\vec{p}_k\right)$ is bigger if $\vec{d}_k$ and $\vec{\nabla} f_k$ are aligned. This means that, if $\alpha_k > 0$, the Steepest Descent is along the direction $-\vec{\nabla} f_k$. Visually, the Steepest Descent step models a ball rolling down following the shape of the graph. This method is simple because only needs the computation of the gradient $\vec{\nabla} f_k$ but it can be "excruciatingly slow on difficult problems" [13], as in the example of Figure 3.1 where the "iterates zigzag toward the solution" [13].

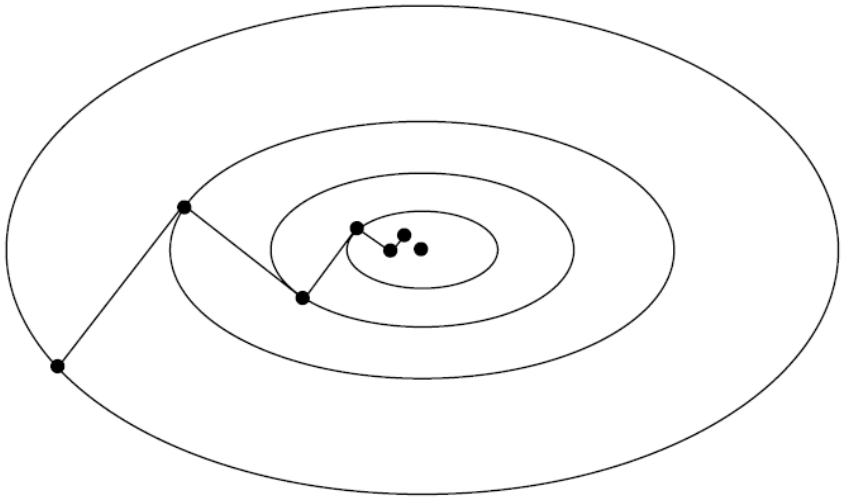


Figure 3.1: Steepest Descent steps [13].

This is because the Steepest Descent steps are perpendicular to the contour lines, so the optimization is completely local. Maybe, the Steepest Descent direction is not the optimal direction to get the solution.

3.1.2 Newton method

The most important line search method, is the one that takes the Newton direction. In this case, the Taylor series with also two terms is taken from $\vec{F}$

$$\vec{F}\left(\vec{p}_k + \vec{d}_k\right) \simeq \vec{F}\left(\vec{p}_k\right) + J\left(\vec{p}_k\right)\, \vec{d}_k, \tag{3.6}$$

where $J(\vec{p}_k)$ is the Jacobian matrix evaluated at the current iteration $\vec{p}_k$. The goal is to get $\vec{F}\left(\vec{p}_k + \vec{d}_k\right) = 0$, so the Newton direction is

$$\vec{d}_k^N = -J\left(\vec{p}_k\right)^{-1} \vec{F}\left(\vec{p}_k\right). \tag{3.7}$$

By construction, if the direction $\vec{d}_k^N$ is fixed, the corresponding line search method (3.3) always gets a unit step $\alpha_k = 1$. Compared to the Steepest Descent method, the Newton step solves a more global problem and, in practice, the solution is reached before. The speed of this method can be observed in Figure 3.2..

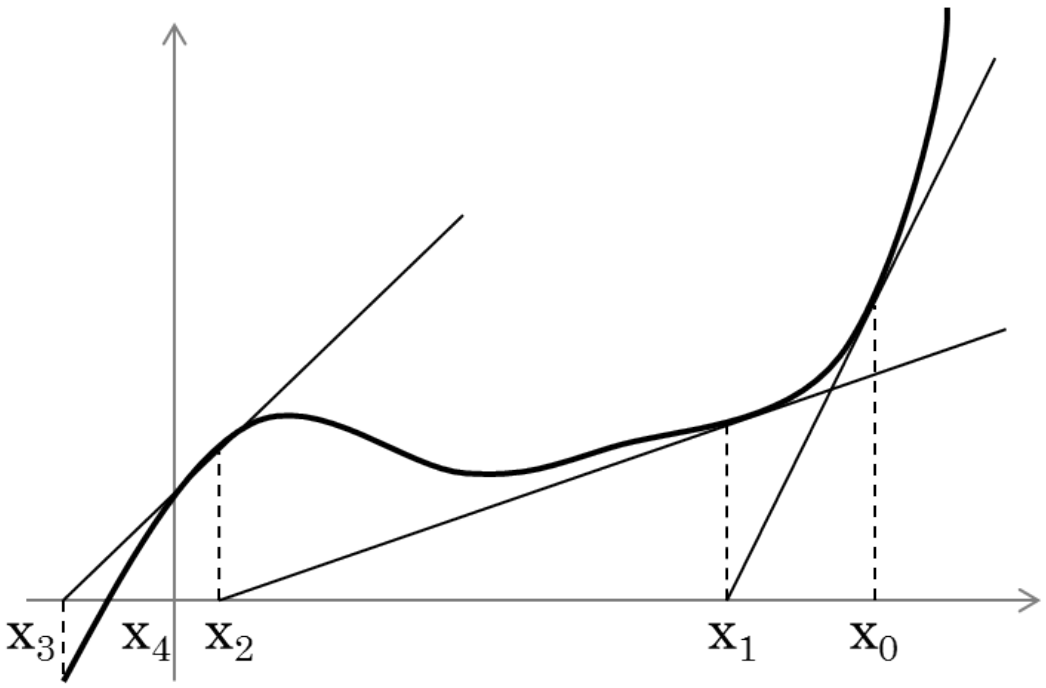


Figure 3.2: 1D example of the Newton method.

Nevertheless, if the Jacobian matrix $J(\vec{p}_k)$ cannot be inverted, there is no solution for the Newton step and the method fails. As well it can happen that the method diverges if $J(\vec{p}_k)$ is close to singular. So the Newton method does not always get a solution. For this reason, the Trust Region methods are used.

3.1.3 Dogleg method

In order to get an algorithm more robust, the Trust Region methods are used because they are adapted to the performance of the solver. The trust region is a neighborhood of the current best solution $\vec{p}_k$ where an approximation m_k of the objective function f reasonably reflects the behaviour of f, locally.

The Dogleg method proposes to use the following convex quadratic approximation of f in the surrounding of $\vec{p}_k$

$$m_k\left(\vec{d}_k\right) = \frac{1}{2}\left\|\vec{F}\left(\vec{p}_k\right) + J\left(\vec{p}_k\right)\vec{d}_k\right\|^2 = \frac{1}{2}\vec{F}_k^T\vec{F}_k + \vec{F}_k^T J\left(\vec{p}_k\right)\vec{d}_k + \frac{1}{2}\vec{d}_k^T J^T\left(\vec{p}_k\right) J\left(\vec{p}_k\right)\vec{d}_k, \tag{3.8}$$

because the Newton step is a root of it $m_k\left(\vec{d}_k^N\right) = 0$. This new objective function m_k should not be confused with the mass flow rate between two contiguous rooms $\dot{m}_k$.

Using the definition of f (3.1) and the expression of its derivative $\nabla\vec{f}_k$ (3.2), the objective function m_k (3.8) looks similar to the Taylor expansion in (3.5) but with an added term

$$m_k\left(\vec{d}_k\right) = f\left(\vec{p}_k\right) + \vec{\nabla}^T f_k\,\vec{d}_k + \frac{1}{2}\vec{d}_k^T J^T\left(\vec{p}_k\right) J\left(\vec{p}_k\right)\vec{d}_k. \tag{3.9}$$

This third term, instead of having the Hessian matrix as the Taylor series does, uses the matrix $J^T\left(x_k\right) J\left(x_k\right)$ that is symmetric and positive-definite. This matrix ensures the convexity of the local approximation of f.

In Figure 3.3, there is a comparison between applying the Trust Region method (3.4) with f (solid lines) and the Newton step using the new objective function (3.8) (dashed lines). Getting the solution of m_k is easier than using f because the contour lines are concentric and they take you directly to the solution.

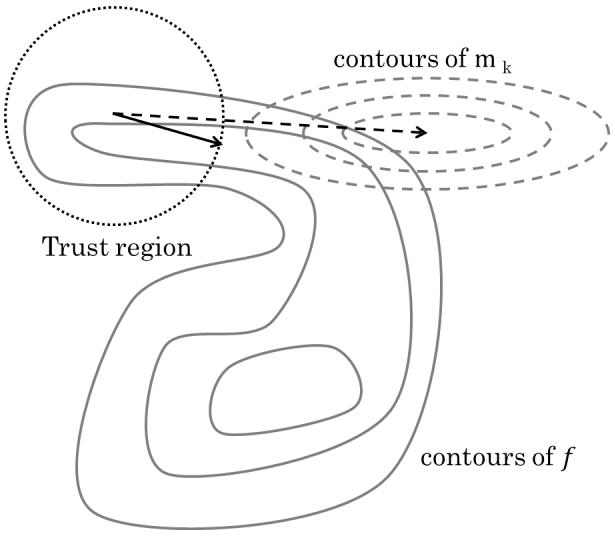


Figure 3.3: Comparison of objective functions f and m_k, based on [13].

If the Jacobian matrix $J(\vec{p}_k)$ is close to singular, the Newton step will blow up. Then, in order to get an accurate solution, the trust region will adapt to the situation reducing its size. If the trust region is tiny, the trial directions are small so the third term in (3.8) can be neglected. Then, the Steepest Descent method is recovered, because the first two terms correspond to (3.5). Using the objective function m_k, the minimization problem corresponding to the Steepest Descent method is

$$m_k\left(-\alpha_k \vec{\nabla} f_k\right) = f\left(\vec{p}_k\right) - \vec{F}_k^T J\left(\vec{p}_k\right) \vec{\nabla} f_k \alpha_k + \frac{1}{2} \vec{\nabla} f_k^T J^T\left(\vec{p}_k\right) J\left(\vec{p}_k\right) \vec{\nabla} f_k \alpha_k^2. \tag{3.10}$$

As said before, line search methods are one-dimensional problems, so the solution of the optimization problem is

$$\alpha_k = \frac{\vec{F}_k^T J\left(\vec{p}_k\right) \vec{\nabla} f_k}{\vec{\nabla} f_k^T J^T\left(\vec{p}_k\right) J\left(\vec{p}_k\right) \vec{\nabla} f_k} = \frac{\vec{F}_k^T J\left(\vec{p}_k\right) J^T\left(\vec{p}_k\right) \vec{F}_k}{\vec{F}_k^T J\left(\vec{p}_k\right) J^T\left(\vec{p}_k\right) J\left(\vec{p}_k\right) J^T\left(\vec{p}_k\right) \vec{F}_k} = \frac{\left\| J^T\left(\vec{p}_k\right) \vec{F}_k \right\|^2}{\left\| J\left(\vec{p}_k\right) J^T\left(\vec{p}_k\right) \vec{F}_k \right\|^2}. \tag{3.11}$$

Then, replacing α_k in the Steepest Descent direction, it is obtained

$$\vec{d}_k^C = -\frac{\left\| J^T\left(\vec{p}_k\right) \vec{F}_k \right\|^2}{\left\| J\left(\vec{p}_k\right) J^T\left(\vec{p}_k\right) \vec{F}_k \right\|^2} \vec{\nabla} f_k, \tag{3.12}$$

the Cauchy step, which is the Steepest Descent step corresponding to the objective function m_k. The geometric interpretation of this step can be observed in Figure 3.4.

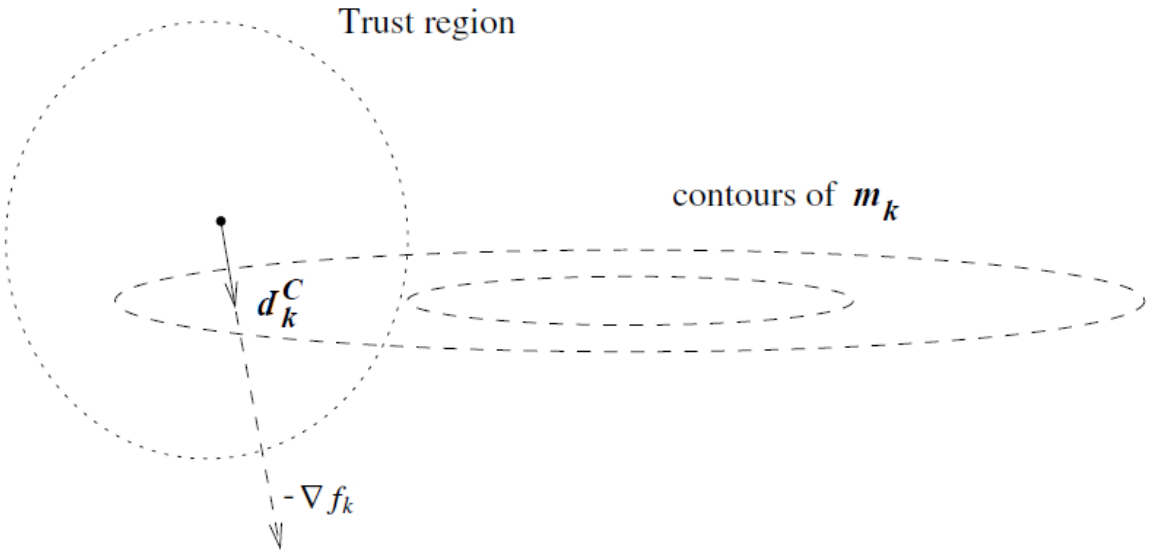


Figure 3.4: Cauchy step [13].

Once the Cauchy step is computed, in order to get the best of both Newton (3.7) and Cauchy (3.12) steps, the Dogleg method proposes to use a linear combination of them. In Figure 3.5 there is the flow chart of the algorithm for choosing the Dogleg step.

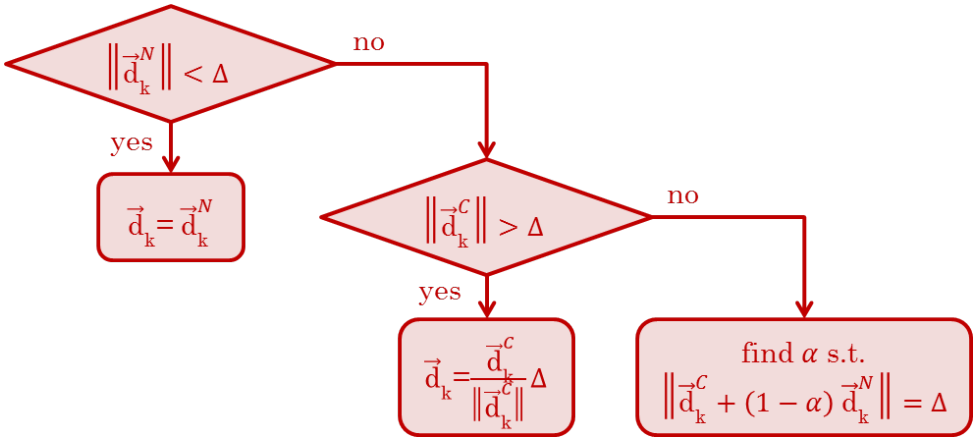


Figure 3.5: Decision tree to get the Dogleg step.

If the trust region is large enough, then the Newton step is the best choice. But if the trust region is tiny, the Steepest Descent method must be used to ensure the existence of a solution.

The geometric interpretation of this Dogleg step is drawn in Figure 3.6 where first, the Dogleg step tries to follow the Steepest Descent direction, but if the trust region is large enough, the Dogleg step approaches to the Newton step (full step).

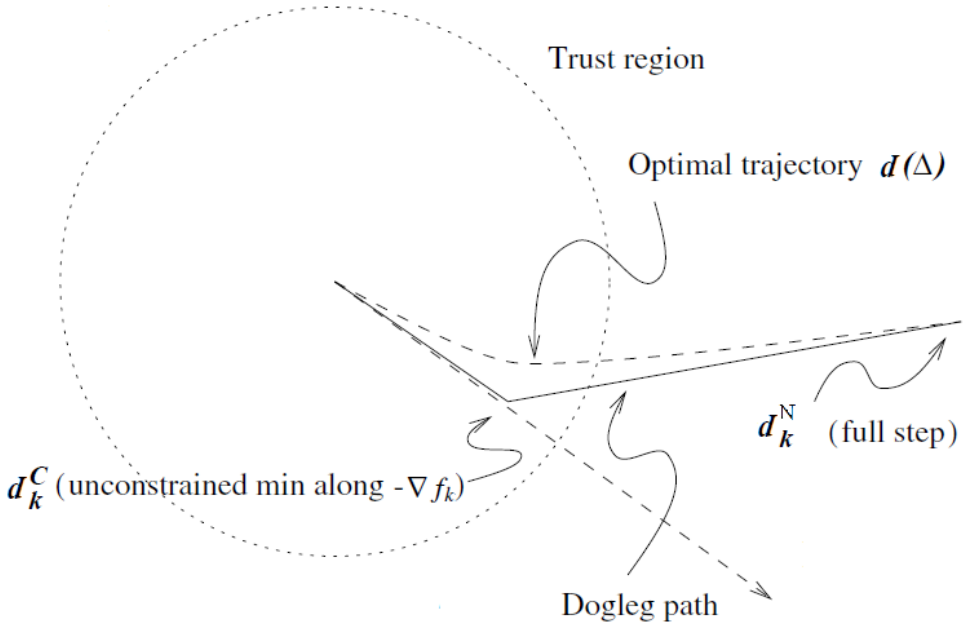


Figure 3.6: Exact trajectory and Dogleg approximation [13].

The Dogleg method is used in MATLAB by default, because is robust an efficient since it only requires one linear solver per iteration, for the computation of the Newton step [11]. The Dogleg method is the winner of the struggle between accuracy (Steepest Descent method) and computational cost (Newton method).

3.2 Jacobian matrix

In this Section, the Jacobian matrix is going to be computed because it appears in all the methods described above. In this work, the computation of the Jacobian matrix is easy because the derivative of the mass flow rate with respect to the pressure of a contiguous room follows the power rule. Otherwise, depending on the distribution of the house or the size of the building, the Jacobian matrix can be large and dense, so problems of memory can appear. To address this problem, a Jacobian-free approach is proposed by Knoll and Wilson in [8].

3.2.1 NEST-Building Jacobian matrix

The Jacobian matrix is defined as $J_{ij} = \dfrac{\partial F_i}{\partial p_j}$ where F_i corresponds to the mass balance of the current room i and p_j to the pressure of room j

$$F_i\left(\vec{p}\right) = \sum_{k \in Inlet} \dot{m}_k\left(p_k^+, p_i\right) - \sum_{k \in Outlet} \dot{m}_k\left(p_i, p_k^-\right). \tag{3.13}$$

Every opening k connects a room at higher pressure $p_k^+ = \max\left(p_i, p_j\right)$ with another that has lower pressure $p_k^- = \min\left(p_i, p_j\right)$. Whether p_i is one or the other, the opening k is considered as an inlet or as an outlet. If $p_i = p_j$, there is no mass flow between the rooms.

In this work, there is a bijection between neigbhour rooms j and openings $k(j)$, because if there are two openings that connect the same two rooms, the areas are added in order to get a unique opening. Then, in an outlet $p_{k(j)}^+ = p_j$, while in an inlet $p_{k(j)}^- = p_j$.

If there is an opening between i and j, the partial derivative of F_i with respect to the pressure of the contiguous room p_j is

$$\frac{\partial F_i}{\partial p_j} = \operatorname{sgn}\left(p_j - p_i\right) \frac{\partial \dot{m}_{k(j)}}{\partial p_j} \quad \forall j \neq i, \tag{3.14}$$

because there is only one linking opening. The sign depends on whether the corresponding opening $k(j)$ is considered as an inlet or as an outlet in (3.13). Now, developing the partial derivative of the mass flow rate (2.7)

$$\frac{\partial F_i}{\partial p_j} = \operatorname{sgn}\left(p_j - p_i\right) \operatorname{sgn}\left(p_j - p_i\right) C_d A \sqrt{\frac{\rho_{k(j)}^+}{2\left(p_{k(j)}^+ - p_{k(j)}^-\right)}} \quad \forall j \neq i, \tag{3.15}$$

because the sign of $\dfrac{\partial \dot{m}_{k(j)}}{\partial p_j}$ depends also if the room j is situated downwards from the opening or upwards. This means that the partial derivative is positive because $\operatorname{sgn}^2\left(p_j - p_i\right) = 1$.

On the other hand, the derivative with respect to the pressure of the room i is

$$\frac{\partial F_i}{\partial p_i} = \sum_{k \in Inlet} \frac{\partial \dot{m}_k}{\partial p_i} - \sum_{k \in Outlet} \frac{\partial \dot{m}_k}{\partial p_i}, \tag{3.16}$$

because the current room i contributes to the mass flow rate of all the contiguous neighbours. But, thanks to the bijection, the loop can be done through the contiguous rooms j

$$\frac{\partial F_i}{\partial p_i} = \sum_j \text{sgn}\left(p_j - p_i\right) \frac{\partial \dot{m}_{k(j)}}{\partial p_i}. \tag{3.17}$$

The partial derivative of $\dot{m}_k$, with respect to p_i, has the opposite sign when compared to the derivative with respect to p_j because the upward/downward rooms are exchanged

$$\frac{\partial F_i}{\partial p_i} = \sum_j \text{sgn}\left(p_j - p_i\right) \text{sgn}\left(p_i - p_j\right) C_d A \sqrt{\frac{\rho^+_{k(j)}}{2\left(p^+_{k(j)} - p^-_{k(j)}\right)}}. \tag{3.18}$$

This means that all the terms of the summation are negative, in comparison with (3.15), because $\text{sgn}\left(p_j - p_i\right) \text{sgn}\left(p_i - p_j\right) = -1$. Hence, the partial derivatives (3.15) and (3.18) are related

$$\frac{\partial F_i}{\partial p_i} = -\sum_{j \neq i} \frac{\partial F_i}{\partial p_j}, \tag{3.19}$$

if the room is interior. If the room is exterior, the relation becomes an inequality because the outside pressure is constant, $\dfrac{\partial F_i}{\partial p_{Outside}} = 0$. Then, the Jacobian matrix $J_{ij} = \dfrac{\partial F_i}{\partial p_j}$ is diagonally dominant

$$|J_{ii}| = -\frac{\partial F_i}{\partial p_i} = \sum_{j \neq i} \frac{\partial F_i}{\partial p_j} \geq \sum_{j \neq i} \left|\frac{\partial F_i}{\partial p_j}\right| = \sum_{j \neq i} |J_{ij}| \quad \forall i, \tag{3.20}$$

because the diagonal elements are negative and the non-diagonal are positive. In fact, the rows of the Jacobian corresponding to the outside *Rooms*, are the ones that are strictly dominant.

A diagonally dominant matrix with negative diagonal entries is negative definite if it is also symmetric. It is clearly the case because

$$\frac{\partial F_i}{\partial p_j} = C_d A \sqrt{\frac{\rho^+_{k(j)}}{2\left(p^+_{k(j)} - p^-_{k(j)}\right)}} = C_d A \sqrt{\frac{\rho^+_{\bar{k}(i)}}{2\left(p^+_{\bar{k}(i)} - p^-_{\bar{k}(i)}\right)}} = \frac{\partial F_j}{\partial p_i}, \tag{3.21}$$

where k is an opening of room i and $\bar{k}$ belongs to room j.

In summary, the Jacobian matrix of this problem is symmetric, diagonal dominant with all the diagonal entries negatives and, consequently, negative definite. This will enable to use the Conjugate Gradient method in the Newton step, reducing the complexity of the linear solver and improving, consequently, the performance of the Trust Region Dogleg method. Otherwise, may be, the connectivity of the room implies a really large and dense Jacobian matrix that makes the method impracticable due to problems of memory. For this reason, in the next Sub-Section, a Jacobian-free approach is introduced, in order to have an alternative to the analytical computation of the Jacobian matrix J_{ij}.

3.2.2 Jacobian-free method

Every time that the Jacobian matrix appears in this work, it implies a matrix-vector product. This Jacobian-vector product corresponds to a directional derivative [13]

$$J\left(\vec{p}\right)\,\vec{q} = \lim_{\varepsilon \to 0} \frac{\vec{F}\left(\vec{p} + \varepsilon\,\vec{q}\right) - \vec{F}\left(\vec{p}\right)}{\varepsilon}. \tag{3.22}$$

The Jacobian-free methods use an approximation of this directional derivative. In this work, a second-order approximation is constructed, using central-differences [8]

$$J\left(\vec{p}\right)\,\vec{q} \simeq \frac{\vec{F}\left(\vec{p} + \varepsilon\,\vec{q}\right) - \vec{F}\left(\vec{p} - \varepsilon\,\vec{q}\right)}{2\varepsilon}. \tag{3.23}$$

To be accurate, ε must be small enough, but if it is too small "the result of the finite difference is contaminated by floating-point round-off error" [8]. To avoid this error, the choice of ε is [8]

$$\varepsilon = \frac{\sqrt{\left(1 + \|\vec{p}\|\right)\varepsilon_{\text{mach}}}}{\|\vec{q}\|}, \tag{3.24}$$

where $\varepsilon_{\text{mach}}$ is the machine epsilon, an upper bound on the relative error due to rounding in floating-point. Then, instead of saving completely a large matrix, the Jacobian-free method lets reduce the memory cost by doing only matrix-vector operations. Otherwise, the evaluation of $\vec{F}$ is not automatic. For this reason, the Jacobian-free method will be compared with computing directly the Jacobian matrix.

3.3 Trust Region Dogleg method

At this point of the work, the properties of the system of equations have been studied and the basic ideas of different non-linear solvers have been introduced. In this Section, the Trust Region Dogleg method is going to be explained in depth.

The key point of the Trust Region methods is the update of the size Δ_k of the trust region. If the current iteration makes a satisfactory reduction, the trust region can be expanded setting a larger Δ_k in the next iteration. If only is achieved a limited improvement of the solution, the trust region should be maintained, or in the worst cases, the trust region must be shrunk. The adaptation of the trust region depends on the ratio of the actual reduction to the predicted reduction

$$\rho_k = \frac{f\left(\vec{p}_k\right) - f\left(\vec{p}_k + \vec{d}_k\right)}{m_k\left(\vec{0}\right) - m_k\left(\vec{d}_k\right)}. \tag{3.25}$$

This gives an estimation of the performance of the algorithm. If $\rho_k \simeq 1$ the quadratic function m_k is a good approximation of f so the trust region can be expanded. If $\rho_k \leq 0$ the approximation is far from f and the trust region must be reduced. Then, the Trust Region algorithm is the following

Algorithm 1 Trust Region

Set $k := 0$, $\vec{p}_0 := \vec{0}$ and $\Delta_0 := 1$
while $\left\|\vec{F}(\vec{p}_k)\right\| > 0$ and $\left\|\vec{d}_k\right\| > 0$ **do**
    Compute the Dogleg step $\vec{d}_k$ using Figure 3.5 and evaluate ρ_k using (3.25)
    if $\rho_k < 1/4$ **then**
        $\Delta_{k+1} := \Delta_k/4$
    else if $\rho_k > 3/4$ and $\left\|\vec{d}_k\right\| = \Delta_k$ **then**
        $\Delta_{k+1} := 2\,\Delta_k$
    end if
    if $\rho_k > 1/10000$ **then**
        $\vec{p}_{k+1} := \vec{p}_k + \vec{d}_k$
    end if
    $k := k + 1$
end while

This algorithm has been taken from Nocedal and Wright [13] and the default parameters are taken from the WOLFRAM software [20]. Initially the rooms of the building are considered at zero pressure because there is not enough information but they will not be far from the atmospheric pressure. In Figure 3.7 there is an example of the evolution of the trust region during an optimization process.

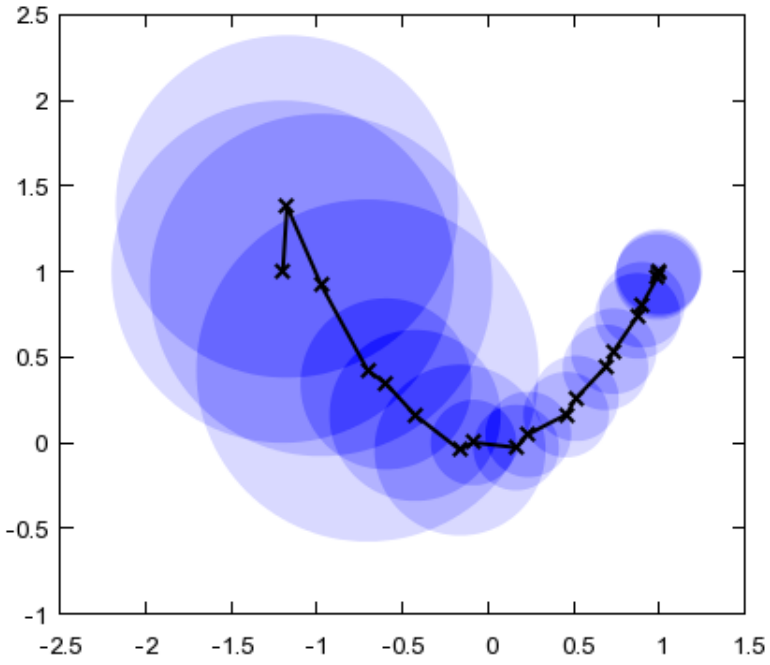


Figure 3.7: Example of a Trust Region method optimization trajectory [21].

The previous pseudo-code of the Trust Region method described in Algorithm 1 is focused on the generalized part of the minimization problem, because the third line of the code can be run using any other type of step. In this work, the Dogleg step is used

Algorithm 2 Dogleg method

1: Compute the Cauchy step $\vec{d}_k^C$ using (3.12) and the Newton step $\vec{d}_k^N$ using (3.7)
2: **if** $\left\|\vec{d}_k^N\right\| < \Delta$ **then**
3: $\quad \vec{d}_k := \vec{d}_k^N$
4: **else if** $\left\|\vec{d}_k^C\right\| > \Delta_k$ **then**
5: $\quad \vec{d}_k := \dfrac{\vec{d}_k^C}{\left\|\vec{d}_k^C\right\|}\Delta_k$
6: **else**
7: $\quad$ Find λ_k such that $g(\lambda_k) = \left\|\vec{d}_k^C + (1-\lambda_k)\,\vec{d}_k^N\right\| - \Delta_k = 0$ using Brent's method
8: $\quad \vec{d}_k := \vec{d}_k^C + (1-\lambda_k)\,\vec{d}_k^N$
9: **end if**

Computing the Cauchy step is easy because it only involves basic matrix-vector operations and the Newton step needs to solve a linear system. But if the Dogleg step consists on a linear combination of the Steepest Descent method and the Newton step, the point of intersection with the trust region must be found. To do so, the Brent's method is used, that combines inverse quadratic interpolation, the bisection method and the secant method for root finding [17]. As the Dogleg method in optimization, the Brent's algorithm is a combination of the best aspects of these different root finders.

The secant method is the root-finding version of the Newton method, while the bisection method is a divide and conquer algorithm that ensures the convergence of the method, as the Steepest Descent method does in optimization. The Brent's method makes a step further and instead of considering a linear approximation, as the secant method does, it uses a Lagrange interpolating polynomial of degre 2. If the Lagrange polynomials cannot be computed, the secant method is used.

The Brent's method begins with two points a_0 and b_0, such that their evaluation have opposite sign $g(a_0) \cdot g(b_0) < 0$, in order to ensure the existence of a root. This is the case of the Dogleg algorithm because $g(1) = \left\|\vec{d}_k^C\right\| - \Delta_k < 0$ and $g(0) = \left\|\vec{d}_k^N\right\| - \Delta_k > 0$ when the Brent's method is run. In order to do the polynomial interpolation, the algorithm needs three points: the point b_k is considered a guess for the root of g, a_k is a worse guess for the the root such that $g(a_k) \cdot g(b_k) < 0$ and b_{k-1} is the previous iterate ($b_{-1} = a_0$).

Algorithm 3 Brent's method

if $|g(a)| < |g(b)|$ **then**
  Swap a and b
end if
$c := a$
while $|g(b)| > 0$ and $|a - b| > 0$ **do**
  if $|g(a)| \neq |g(c)|$ and $|g(b)| \neq |g(c)|$ **then**
    (inverse quadratic approximation)
    $$s := \frac{a\,g(b)\,g(c)}{(g(a)-g(b))(g(a)-g(c))} + \frac{b\,g(a)\,g(c)}{(g(b)-g(a))(g(b)-g(c))} + \frac{c\,g(a)\,g(b)}{(g(c)-g(a))(g(c)-g(b))}$$
  else
    $$s := b - g(b)\frac{b-a}{g(b)-g(a)}$$ (secant method)
  end if
  if s if not between $\frac{3a+b}{b}$ and b **then**
    $$s := \frac{a+b}{2}$$ (bisection method)
  end if
  $c := b$ (the previous iterate)
  if $g(a) \cdot g(s) < 0$ **then**
    $b := s$
  else
    $a := s$
  end if
  if $|g(a)| < |g(b)|$ **then**
    Swap a and b (b is the current guess of the root)
  end if
end while

As the Dogleg method in optimization, the Brent's method is the winner of the struggle between accuracy (bisection method) and computational cost (secant method). The bisection method has logarithmic complexity, but it is used only when the secant method blows up. In addition, using an inverse quadratic approximation, instead of the linear approximation of the secant method, makes the algorithm converge to the solution faster.

Now, all that remains is to explain linear solver needed to compute the Newton step. As the Jacobian matrix of the problem is symmetric negative-definite 3.2, the Conjugate Gradient method is used. It is adapted in order to become a Jacobian-free method, avoiding the computation of the Jacobian matrix.

3.3.1 Conjugate Gradient method

The Newton step (3.7) consists on solving the system of linear equations

$$-J\left(\vec{p}_k\right)\vec{d}_k^N = \vec{F}\left(\vec{p}_k\right), \tag{3.26}$$

where the Jacobian matrix is symmetric and negative definite, as already seen in Sub-Section 3.2.1. Then, the Newton step consists on a linear system of equations with a positive definite matrix. Otherwise, instead of solving (3.26), the conjugate gradient method optimizes the quadratic function

$$q_k\left(\vec{d}\right) = -\frac{1}{2}\vec{d}^T J\left(\vec{p}_k\right)\vec{d} - \vec{d}^T\vec{F}\left(\vec{p}_k\right), \tag{3.27}$$

because a minimum of the quadratic function (3.27) corresponds to a solution of the linear system (3.26).

In order to reach a minimum of (3.27), the conjugate gradient follows the Steepest Descent direction in every iteration

$$-\nabla q_k\left(\vec{d}_i\right) = J\left(\vec{p}_k\right)\vec{d}_i + \vec{F}\left(\vec{p}_k\right) = \vec{r}_i, \tag{3.28}$$

that is equal to the residual of the linear system (3.26). The minimum exists because, as the matrix $-J\left(\vec{p}_k\right)$ is positive definite, the quadratic form (3.27) corresponds to a convex function.

But the residual directions must be corrected because the Conjugate Gradient method uses an orthogonal basis. The Gramm-Schmidt process orthogonalizes the directions $\vec{\delta}_i$ with respect to the inner product defined by the Jacobian matrix

$$\vec{\delta}_i = \vec{r}_i - \sum_{j<i}\frac{\vec{\delta}_j^T J\left(\vec{p}_k\right)\vec{r}_i}{\vec{\delta}_j^T J\left(\vec{p}_k\right)\vec{\delta}_j}\vec{\delta}_j. \tag{3.29}$$

Then, the line search method related to this direction is

$$q_k\left(\alpha_i\,\vec{\delta}_i\right) = -\frac{1}{2}\vec{\delta}_i^T J\left(\vec{p}_k\right)\vec{\delta}_i\alpha_i^2 - \vec{\delta}_i^T\vec{F}\left(\vec{p}_k\right)\alpha_i, \tag{3.30}$$

whose minimum is reached at

$$\alpha_i = -\frac{\vec{\delta}_i^T\vec{F}\left(\vec{p}_k\right)}{\vec{\delta}_i^T J\left(\vec{p}_k\right)\vec{\delta}_i}. \tag{3.31}$$

Otherwise, to simplify the computational cost and storage, the algorithm is simplified taking advantage of the orthogonality of $\vec{r}_{i+1}$ with respect to the directions $\vec{\delta}_j$ for all $j < i$. Therefore, only $\vec{d}_k$, $\vec{\delta}_k$ and $\vec{r}_k$ are needed to compute $\vec{d}_{k+1}$, $\vec{\delta}_{k+1}$ and $\vec{r}_{k+1}$. This reduces the algorithm to only computing one matrix-vector product per iteration, for the computation of α_i.

Algorithm 4 Conjugate gradient

1: Set $\vec{d}_0 := \vec{0}$, $\vec{r}_0 := \vec{F}(\vec{p}_k)$ and $\vec{\delta}_0 := \vec{r}_0$
2: **while** $\|\vec{r}_i\| > 0$ **do**
3: $\quad \alpha_i := -\dfrac{\vec{r}_i^T \vec{r}_i}{\vec{\delta}_i^T J(\vec{p}_k)\, \vec{\delta}_i}$
4: $\quad \vec{d}_{i+1} := \vec{d}_i + \alpha_i \vec{\delta}_i$
5: $\quad \vec{r}_{i+1} := \vec{r}_i + \alpha_i J(\vec{p}_k)\ \vec{\delta}_i$
6: $\quad \beta_i := \dfrac{\vec{r}_{i+1}^T \vec{r}_{i+1}}{\vec{r}_i^T \vec{r}_i}$
7: $\quad \vec{\delta}_{i+1} := \vec{r}_{i+1} + \beta_i \vec{\delta}_i$
8: $\quad i := i + 1$
9: **end while**

Due to the orthogonality of the basis, the Conjugate Gradient method needs at most as iterations as the number of rooms in the building, in order to reach the solution of the linear system of the Newton step. Nevertheless, using preconditioning will allow to reduce the number of iterations. The diagonal preconditioner seems a good choice because the Jacobian matrix is diagonal dominant.

Otherwise, in every step of the Trust Region method, the Jacobian matrix must be computed for a given pressure $\vec{p}_k$. This Jacobian matrix appears in the Cauchy step, in the computation of the ratio ρ_k (3.25) but, above all, in the Newton step because of the Conjugate Gradient method. It appears a lot and it is always applied to different vectors. The matrix-vector product will be expensive if the Jacobian matrix is large, so using a Jacobian-free method should bring a better performance. Nevertheless, computing the approximation of the directional derivative (3.23) is not trivial because a loop in *Openings* is needed.

Then, in order to choose the best non-linear solver for the problem $\vec{F}(\vec{p}) = 0$ in the next Chapter the Trust Region Dogleg method is going to be executed, computing directly the Jacobian matrix in every step or using a Jacobian-free approximation.

4 Results

In this Chapter, the results of this work are going to be shown using RSTUDIO, because it is an open source code and free. But first, a benchmark problem is solved in order to validate the methods proposed in this work and in order to check that the SIMPLE method is well programmed. Then, once the accuracy of the methods is ensured, they are going to be compared by means of the computational time. The objective is to find a non-linear solver faster than the current solver used by the NEST-Building software, the SIMPLE method.

4.1 Benchmark problem

The benchmark problem is based on the test cases used by Fürbringer *et al.* in [6] but, in order to take into account whether the computation of the Jacobian matrix is correct, three rooms will be tested instead of only one. Because if there is only one room, the Jacobian matrix reduces to a single number corresponding to the unique partial derivative, so more rooms are needed to test the Jacobian matrix. The benchmark problem, that combines the effect of thermal buoyancy and wind velocity around the building, is summarized in the next Figure 4.1.

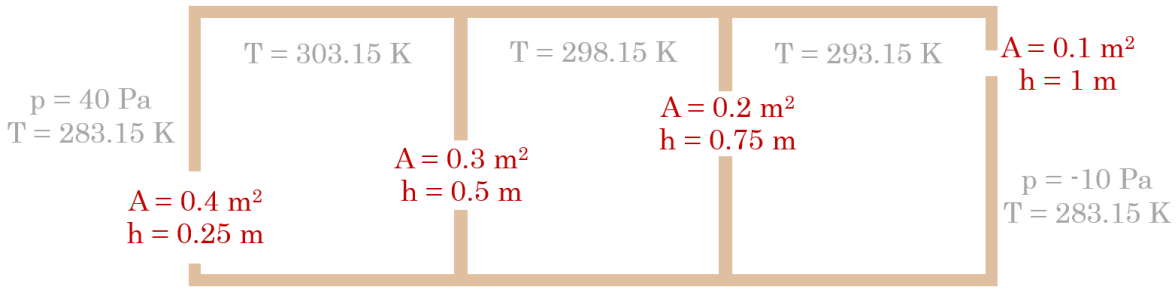


Figure 4.1: Scheme of the benchmark problem.

The problem consists on a system of three equations (one for every mass conservation) and three variables (the pressure of the rooms). The pressure outside the building is considered as boundary conditions. By common sense seems that, if the left wall is at 40 Pa and the right one at -10 Pa, the interior rooms will be at an intermediate pressure, decreasing from left to right. Then, as said during this work, the air flow will follow this gradient of pressure. But, due to mass conservation, the mass flow rate through all the windows should be the same.

Using the expression (2.11) for the mass flow rates and replacing the corresponding parameters from Figure 4.1, the mass flow through the openings is $\dot{m} = 0.325\,96\,\mathrm{m^3\,s^{-1}}$. This corresponds to a pressure of $p = 37.667\,44$ Pa in the left room, $p = 33.707\,41$ Pa in the middle room and $p = 24.880\,94$ Pa in the right one. These values of the pressure are used to check the accuracy of the different methods proposed in this work.

Algorithm	p_{left} [Pa]	p_{middle} [Pa]	p_{right} [Pa]	Error [%]
NEST (SIMPLE)	37.65494	33.67113	24.79201	0.17191
Conjugate Gradient	37.66744	33.70741	24.88094	$6.55225 \cdot 10^{-6}$
Jacobian-free CG	37.66744	33.70741	24.88094	$6.55225 \cdot 10^{-6}$
Preconditioned CG	37.66744	33.70741	24.88094	$6.55225 \cdot 10^{-6}$

Table 4.1: Performance of the algorithms and their error with respect to the analytic solution.

The SIMPLE method, despite being worse than the non-linear solvers proposed in this work, it is accurate. The non-linear solvers bring the same results because they differ by how they compute the linear system corresponding to the Newton step. It will be observed later that this only affects the computational performance, not the accuracy.

The SIMPLE method, in terms of accuracy, is worse because it takes so small steps that when the tolerance is met, the last step is still far from the solution.

4.2 Test case

Once the accuracy of the methods is checked, a comparison of the computational cost can be done. To do so, a test case closer to the reality is needed. The chosen building consists on a cube with $n \times n \times n + 1$ rooms, that allows simulating different building sizes easily. Every room in the cube is linked with its neigbours rooms of the same storey and with the outside, if the room belongs to a façade of the building. An added room at one of the façades is responsible for linking the different storeys, simulating the stairs of the building. In Figure 4.2 there is the building corresponding to $n = 2$.

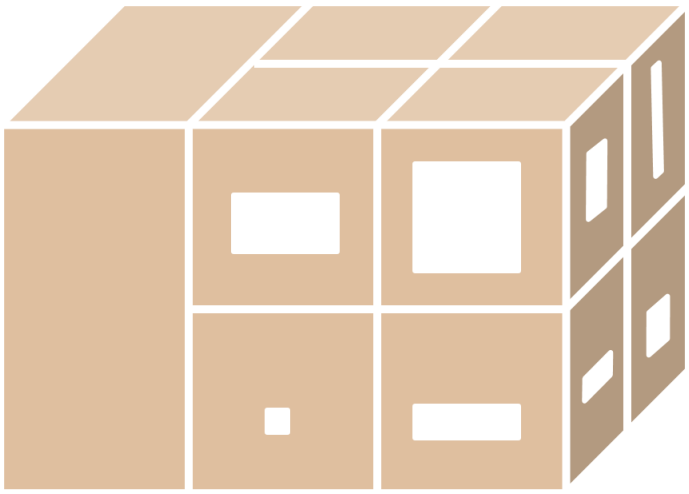


Figure 4.2: Outside of a test case building with $n = 2$.

The size of the windows, as seen in Figure 4.2, varies room-to-room having a maximum width of 2 m and a maximum height of 1 m, so the windows will have a maximum area of 2 m^2. Despite this random distribution of the windows, they are all fixed at 0.9 m from the floor of the room. Then, to know the height of the windows, every storey is considered 2.5 m high floor-to-floor.

This means that the distribution of the building depends on the number of rooms n in each direction of the cube and the size of the windows of the building. In Figure 4.3 it is shown the distribution of the interior of a test case building where, for simplicity, all the doors and windows are considered opened.

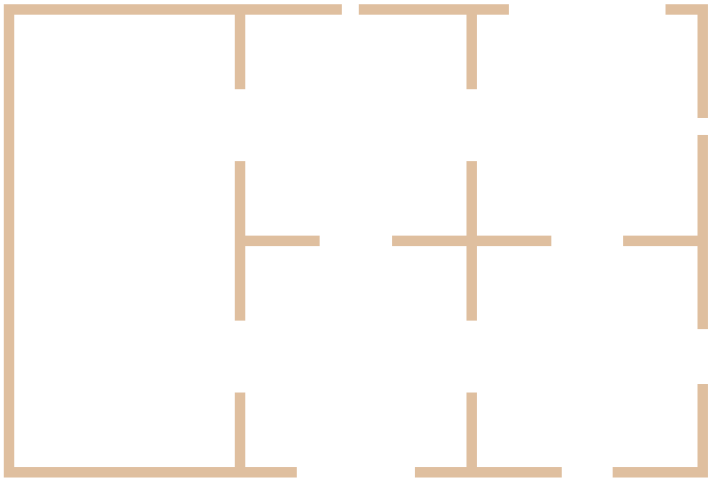


Figure 4.3: One storey of a test case building with $n = 2$.

Unlike windows, doors are standard. This means that they have a width of 0.825 m and a height of 2.11 m, that involves a door area of 1.74 m^2. Of course, doors are placed at floor level.

Apart from the distribution of the building, the weather conditions are also considered as random variables. The temperature of the rooms and the outside follows a uniform distribution between $-10\,°C$ and $40\,°C$ and the outside pressure goes from -50 Pa to 50 Pa, simulating the effect of the wind on the façade. Then, in order to obtain the probability distribution of the computational cost, 1000 Monte Carlo simulations are run, for every number of rooms n, in a Intel® Core ™2 Duo Processor E6700 CPU.

In the benchmark problem, it has been found that the the SIMPLE method takes small steps. This makes reaching convergence difficult, so the SIMPLE algorithm is slow. This behaviour can be observed in Figure 4.4 where, for only 9 and 28 rooms, the SIMPLE performance (violet data) is far from the non-linear solvers (red, green and blue data). It should be note that the time has logarithmic scale, so the computational performance of the SIMPLE and the Trust Region method differ in several orders of magnitude.

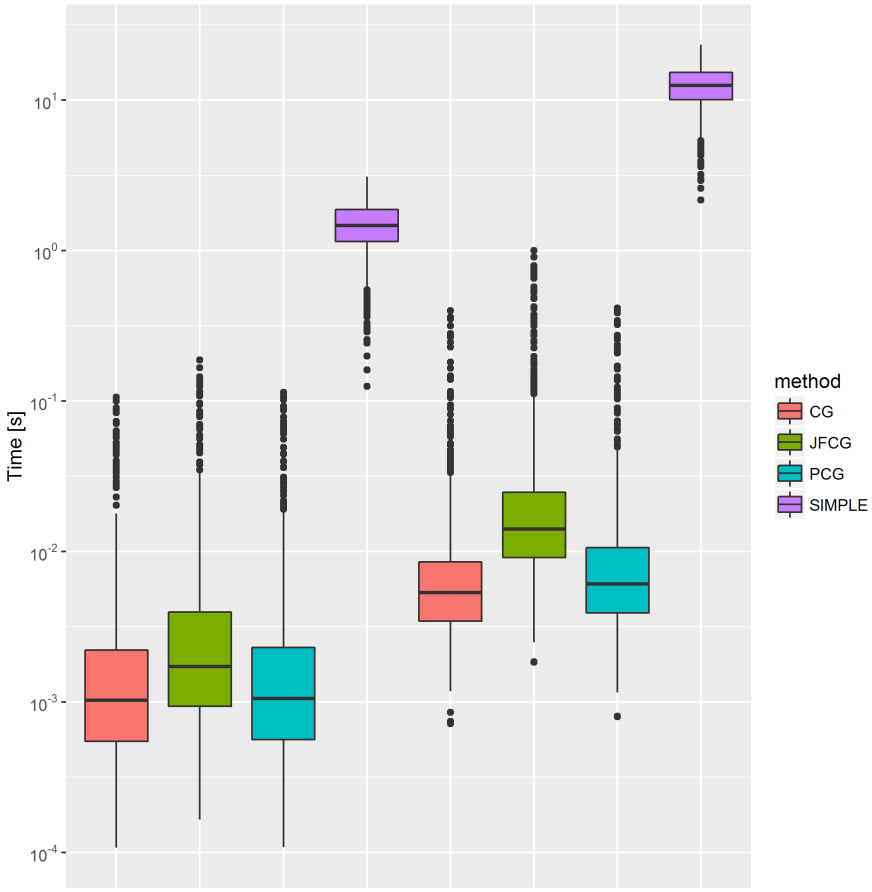
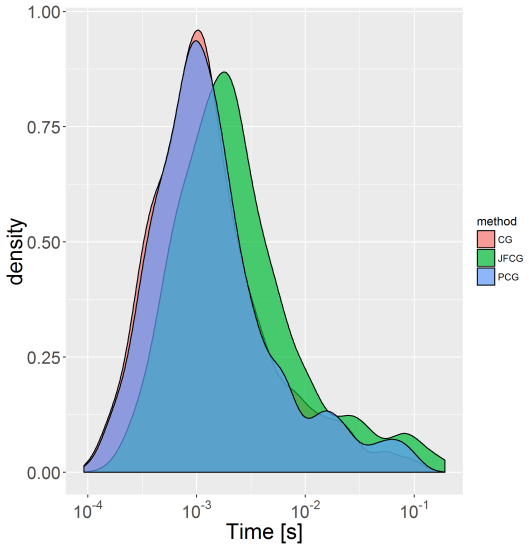


Figure 4.4: Box plot of computational performance for $n = 2$ (left) and $n = 3$ (right).

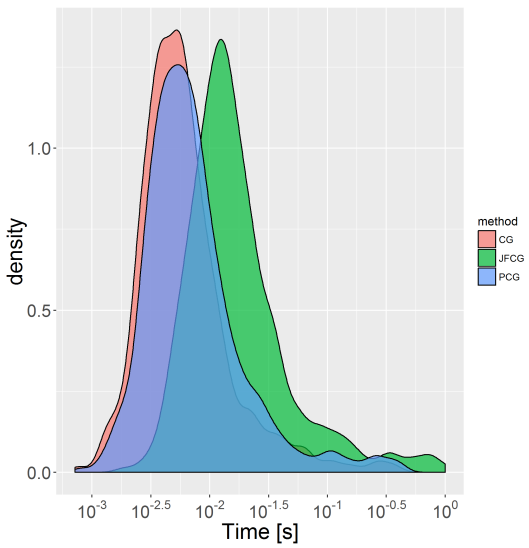
Beyond the horrible performance of the SIMPLE method, it can be observed that there is not much difference between preconditioning (blue data) or not the Conjugate Gradient method (red data) in the Newton step. Otherwise there is difference, but not as much as with the SIMPLE method, between computing the Jacobian matrix directly (blue and red data) or using an approximation of the directional derivative, the Jacobian-free method (green data).

The difference of SIMPLE with the other methods will be bigger if the number of the rooms increases so, from now on, the SIMPLE method will not be simulated and compared with the other methods. Indeed, later will be observed in Figure 4.6 that, while SIMPLE method only solves 9 rooms, the Trust Region methods can solve up to 500 rooms spending the same time.

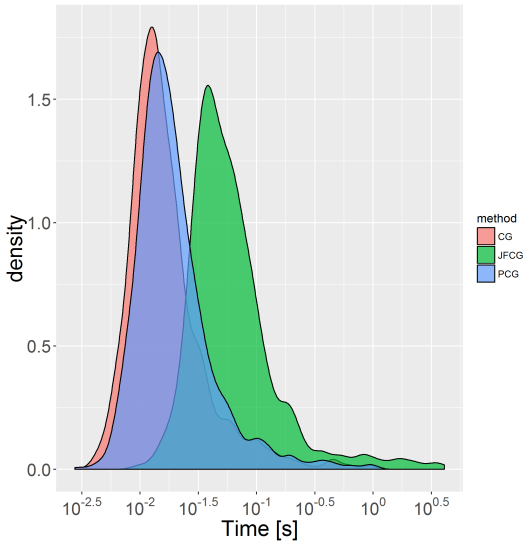
In order to see that the differences between the Trust Region methods grow with the variable n, the probability density function is used instead of a box plot. In the case of preconditioning the linear system, theses differences are smaller, compared to the Jacobian-free method. The simplest Trust Region method (red curve), that solves the linearization by means of the Conjugate Gradient, is gaining ground over the Jacobian-free approach (green curve) and the preconditioned Conjugate Gradient (blue curve), as the number of rooms increases.



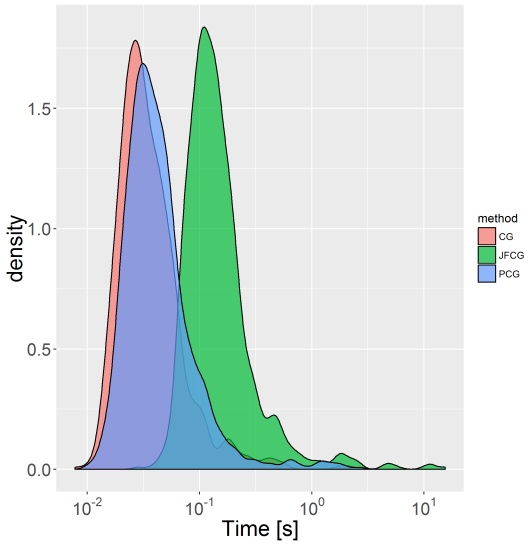
(a) Density functions for $n = 2$ (9 rooms).



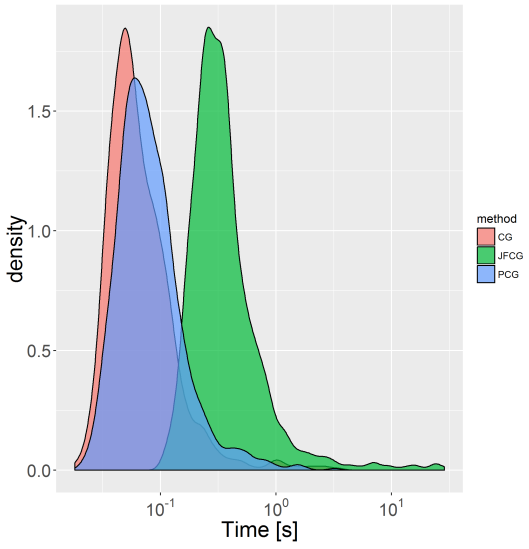
(b) Density functions for $n = 3$ (28 rooms).



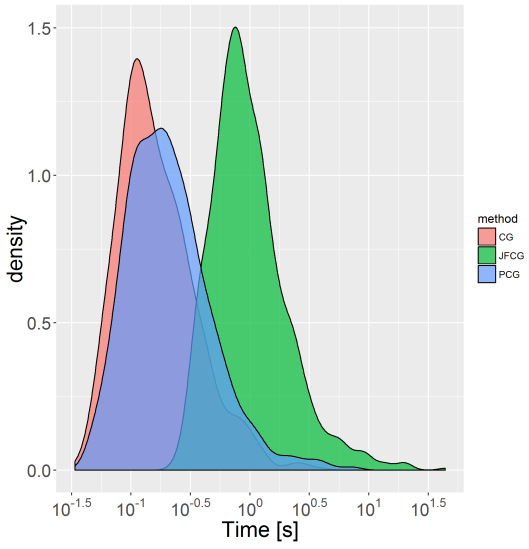
(c) Density functions for $n = 4$ (65 rooms).



(d) Density functions for $n = 5$ (126 rooms).



(e) Density functions for $n = 6$ (217 rooms).



(f) Density functions for $n = 7$ (344 rooms).

Figure 4.5: Probability distributions from $n = 2$ to $n = 7$.

In Figures 4.5 the logarithmic scale is used in order to reject the outliers.

The grow of the computational times with the number of rooms n can be observed better in Figure 4.6 where the different box plots are set up next to each other.

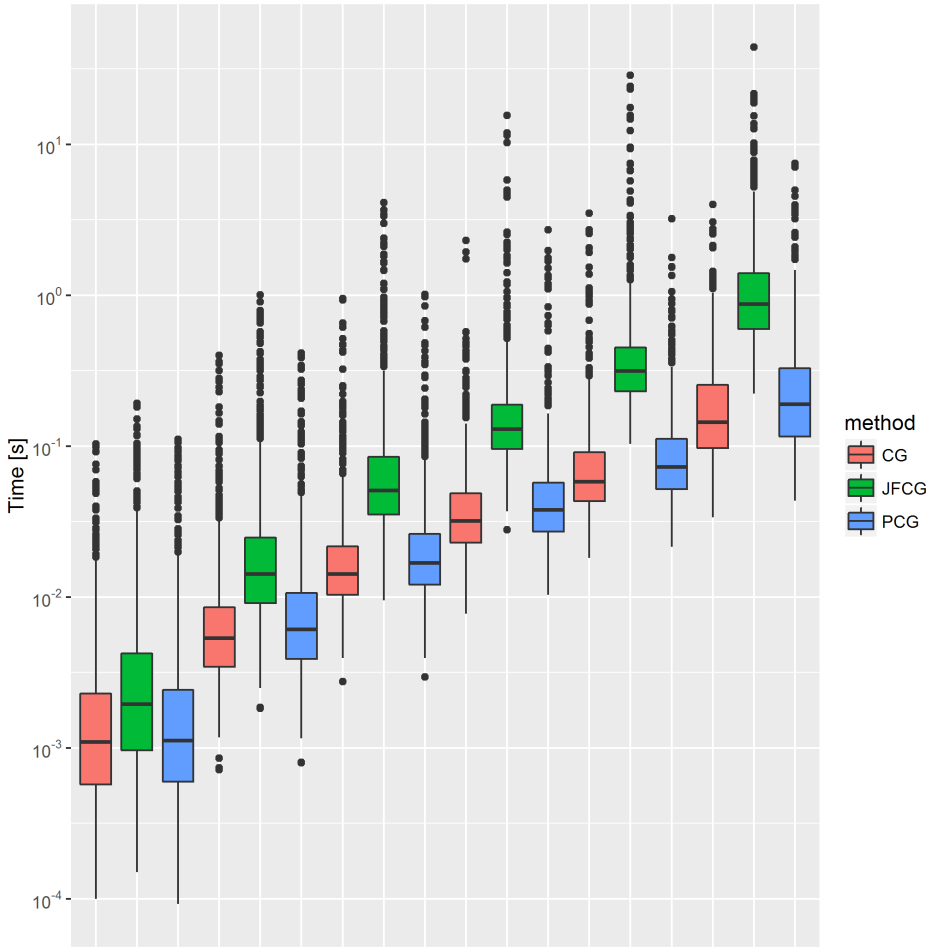


Figure 4.6: Box plots increasing from $n = 2$ (left) to $n = 7$ (right).

4.3 Discussion

Once the results of this work have been introduced, in this Section they are going to be discussed in order to justify them. First of all, must be highlighted that the SIMPLE method has no chance against the Trust Region Dogleg method.

The current solver used by NEST-Building is accurate but is not fast enough to solve the mass conservation in a building. In fact, the SIMPLE method is a simplification of the Newton method but, instead of using the Conjugate Gradient to solve the linear system, the SIMPLE method uses a variation of the Jacobi method. The Jacobi method is an iterative method used in diagonally dominant systems of linear equations, as in the one that is solved in the Newton step (3.7). If the Jacobian matrix is split in diagonal $D(\vec{p}_k)$ and non-diagonal part $R(\vec{p}_k)$, the

Newton step becomes

$$-J\left(\vec{p}_k\right)\vec{d}_k^N = -\left(D\left(\vec{p}_k\right)+R\left(\vec{p}_k\right)\right)\vec{d}_k^N = \vec{F}\left(\vec{p}_k\right). \tag{4.1}$$

So the Jacobi method can be obtained isolating the step like this

$$\vec{d}_k^N = -D\left(\vec{p}_k\right)^{-1}\left(\vec{F}\left(\vec{p}_k\right)+R\left(\vec{p}_k\right)\vec{d}_{k-1}^N\right), \tag{4.2}$$

reducing the Newton step to an iterative method different from the Conjugate Gradient. Otherwise, in order to avoid computing the Jacobian matrix and in order to parallelize the *Rooms* objects, the SIMPLE method neglects the non-diagonal part of the Jacobian matrix $R\left(\vec{p}_k\right)$. As stated before in Sub-Section 2.3.2, this leads to a loss of communication between *Rooms* objects of NEST-Building, but a simpler formulation for the correction of pressure is obtained

$$\vec{d}_k^S = -D\left(\vec{p}_k\right)^{-1}\left(\vec{F}\left(\vec{p}_k\right)+\cancel{R\left(\vec{p}_k\right)\vec{d}_{k-1}^S}\right) = -D\left(\vec{p}_k\right)^{-1}\vec{F}\left(\vec{p}_k\right) = \vec{p}'_k. \tag{4.3}$$

This last equation is equivalent to (2.16) because the diagonal of the Jacobian matrix corresponds to the partial derivatives of the mass flow rates with respect to the pressure of the current room (3.16), as the SIMPLE correction does.

With this simplification, the SIMPLE method avoids the computation of the Jacobian matrix but the linear system (3.7) cannot be solved because, as the error cannot be computed without the Jacobian matrix, it is impossible to know when the solution is reached. This makes the correction $\vec{p}'_k$ really small, compared to the Dogleg steps. Then, the conservation of mass and the partial derivatives must be computed in every step, slowing down the method. In addition, neglecting $R\left(\vec{p}_k\right)$, there is no communication between variables and the Jacobi method loses its appeal. Summarizing, the SIMPLE method takes many long iterations because the steps are very small and they are costly because, in every little step, the mass flux and the partial derivatives must be re-computed.

In the case of the Jacobian-free approximation, this method is useful when the Jacobian matrix is complicated to calculate analytically or when the Jacobian matrix requires a lot of memory space. It is not the case of this work because the mass flow rates, despite having a non-linear expression, have a nice expression and its derivatives, if the worst comes to the worst, follow the power rule. In addition, apart from the fact that in a building there are in the order of at most, a thousand of rooms, the Jacobian matrix shall not be so large. Combined with this, the rooms are only linked with their neighbours, so the Jacobian matrices usually follow a sparsity pattern. Indeed, computing the diagonal part of the Jacobian (SIMPLE method) or computing a directional derivative (Jacobian-free methods), they have the same complexity as computing completely the Jacobian matrix, because all of them need a loop in *Openings* of the

building. Despite the computational cost is of the order of the number of *Openings*, computing the Jacobian matrix is, obviously, more costly than computing only the diagonal of the Jacobian or computing a directional derivative. For this reason, it is better to compute the Jacobian matrix only once in each iteration and to reduce the directional derivatives to a Jacobian-vector product because, this matrix-vector products involves a loop in *Rooms* and the number of rooms in a building is almost always smaller than the number of doors and windows. This reduces the cost of the whole Trust Region Dogleg step because it needs a lot of directional derivatives.

Finally, the best methods are the ones that use the Trust Region Dogleg method, solving the linear system corresponding to the Newton step with the Conjugate Gradient method, but only those who compute the Jacobian matrix completely. The unique difference between preconditioning or not the linear system is that, in each iteration, an auxiliary linear correction is applied to the current residual. This linear correction, in this work, is simple because, as the Jacobian matrix diagonally dominant, it consists on inverting a diagonal matrix. In fact, the correction is similar to the one used by the SIMPLE method (4.3) and its purpose is to reduce the number of iterations of the Conjugate Gradient corresponding to the Newton step. Otherwise, because of the nature of the Jacobian matrix, the Conjugate Gradient is as fast that preconditioning the Conjugate Gradient method, despite reducing the number of iterations, does not reverse the computational cost of the auxiliary linear correction.

5 Planning and programming

The objective of this project is to improve the air mass flow solver in NEST-Building. For this reason, the project has been split in different phases where some tasks must be carried out. These phases are summarized in Figure 5.1 and they are described briefly below.

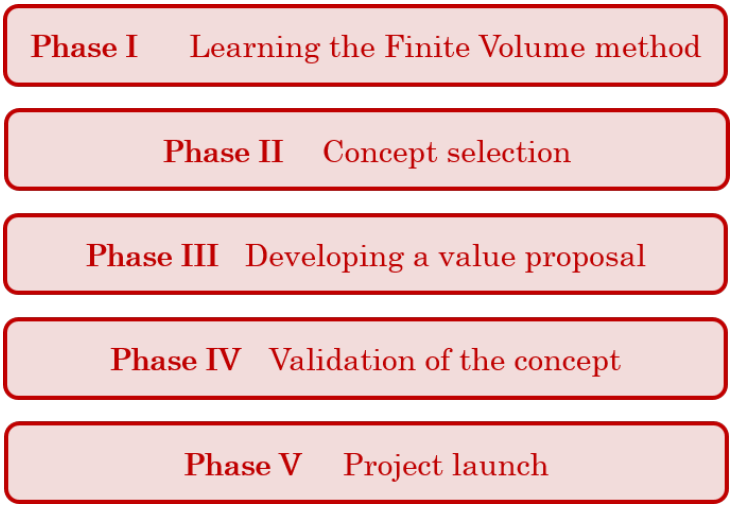


Figure 5.1: Phases of the project.

Phase I: Learning the Finite Volume method

The Heat and Mass Transfer Technological Center (CTTC) is specialized in the Finite Volume method (FVM), a numerical method used for solving PDEs. This method is widely used in CFD and it is the base of the NEST-Building software. Then, it is important to get familiarised with the FVM before going deep to NEST-Building. In CTTC the numerical methods are developed in C++, so having a good background in this programming language is important.

Phase I
Tasks
- Build a C++ environment that works with sparse matrices.
- Build a C++ solver based on the Finite Volume method.
Deliverables
- CFD exercises, from one dimension steady problem to the driven cavity problem.
Objectives
- Familiarise with C++.
- Develop a linear algebra package for C++.
Start 01/07/2016 **Deadline** 30/09/2016

Phase II: Concept selection

When the Finite Volume method and the programming language C++ is mastered, the NES-Building software can be studied in order to define the objectives of the project. Once the weaknesses are found, it is time to propose new solutions exploiting the properties of the particular case in this work. With all this information, the viability of each proposal must be studied in order to find a better solution to the current NEST-Building solver.

Phase II
Tasks
- Study the properties of the model for building air flow.
- Search non-linear solvers.
Deliverables
- A report on the model for building air flow used in NEST-Building.
- A report proposing some non-linear solvers for the mass conservation.
Objectives
- Familiarise with the object-oriented structure used in NEST-Building.
- Propose a new solver for air flow in NEST-Building.
Start 01/10/2016 **Deadline** 15/01/2017

Phase III: Developing a value proposal

Once a better alternative is found for solving the air flow distribution in a building, it must be added to the current NEST-Building software. Due to the fact that in this work a stationary problem has been studied, the solver must be adapted to work in time-dependent problems, as in the case of NEST-Building current solver.

Phase III
Tasks
- Introduce the new solver in the NEST-Building software.
- Adapt the current explicit solver with the new solver as a pressure corrector.
Deliverables
- A report with the proposed amendments.
Objectives
- Improve the performance of the NEST-Building software.
Start 16/01/2017 **Deadline** 28/02/2017

Phase IV: Validation of the concept

This phase includes a verification of the performance of the selected solver in real cases and an adaption of the method to capacities of CTTC. In order to exploit the servers of the laboratory, the code can be parallelised speeding-up the performance of the solver. Finally, as the diagonal preconditioner is not good enough, some other preconditioners should be considered in order to improve the computational cost of the linear solver.

Phase IV			
Tasks			
- Try the new NEST-Building software in real buildings.			
- Study new preconditioners and parallelise the code.			
Deliverables			
- A final report with all the details of the project.			
Objectives			
- Validate the solver in real situations and make it faster.			
Start	01/03/2017	**Deadline**	31/05/2017

Phase V: Project launch

Finally, once the modifications in NEST-Building work effectively, the new capabilities of solver can be introduced definitely in the software of the CTTC. Otherwise, if the improvements of the solver raised in the previous phase have not yet been carried out, this efforts should continue to be. In addition, periodical checks are necessary to see if the implementation is good enough and if it beats the current NEST-Building solver.

Phase V			
Tasks			
- Launch the new software.			
Deliverables			
- A report collecting the incidences of NEST-Building.			
Objectives			
- Obtain a solver faster than the SIMPLE method.			
Start	01/06/2017	**Deadline**	01/09/2017

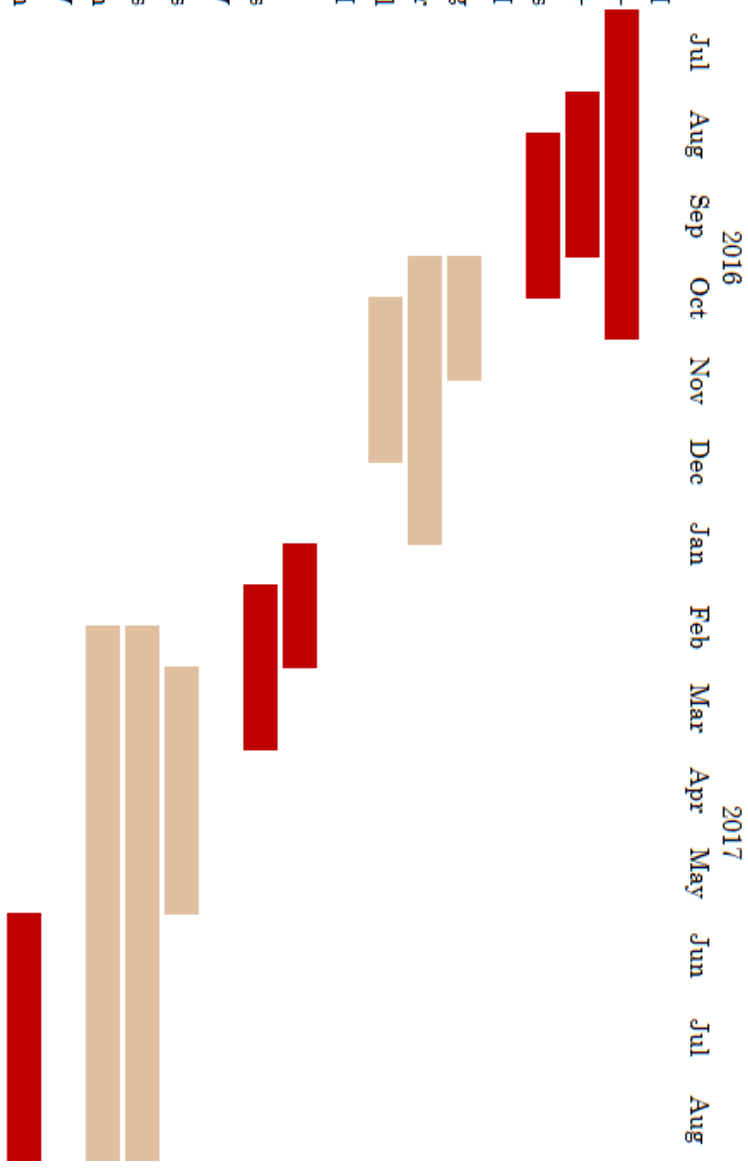


Figure 5.2: Gantt chart with the schedule of the tasks.

6 Budget

In this Chapter, an estimation of the cost of the project is shown. Due to the nature of the project, only the costs related to the human resources have been taken into account. In other words, the costs involved in the time spent by those who have contributed in the elaboration of this project such as scientists or researchers.

This work has been funded by a Banco Santander scholarship, so the cost of the work done by the author is fixed. Nevertheless, other members of CTTC have been involved in this project, so their time spent on consulting information must be added. It has been considered that their work means a 5% of the work done by the author. In 2017, as the scholarship ends, it is considered the one member of the group is going to be dedicated completely to the last three phases of the project. In the wages of the researchers of CTTC, the Social Security taxes of 29.9% are taken into account.

Budget
Author
- Part-time job from 01/07/2016 to 31/12/2016 : 125 days $\times$ 4h = 500 h
- Santander scholarhip: 1,800 €
Consulting
- 500 h $\times$ 5% = 25 h
- 25 h $\times$ 25 €/h $\times$ 1.299 = 811.87 €
Launch
- Full-time from 01/01/2017 to 01/09/2017 : 169 days $\times$ 8 h = 1,352 h
- 1,352 h $\times$ 25 €/h $\times$ 1.299 = 43,906.2 €
Total costs 1,800 + 811.97 + 43,906.2 = 46,518.07 €

7 Environmental impact

The NEST-Building tool is used for energy simulations in buildings. This means that it can affect to the environment in two different ways: the energy consumption is related to the emissions of CO_2 and the air mass flows drive the air quality in a building. Then, it is important to know which are the possible effects of this new solver to the environment and, accordingly, try to minimize the possible negative effects that the project may cause to the environment.

In buildings, the are little openings whose objective it is to control the quality of air inside the rooms but maybe there are some energy losses that must be limited, because these losses are related to an increase of CO_2 emissions. For this reason, it is important to check that the new solver models accurately the air flows in the building and there is no negative impact to the environment.

According to OCU [14], on average, a home consumes 5172 kW h in heating and 170 kW h in air conditioning, during a year. Then, if it is considered that the generation of 1 kW h implies the emission of 0.545 kg of CO_2, the NEST-Building tool can reach a reduction of up to 2911.39 kg of CO_2 per house during a year.

8 Conclusions

In this work, a study of solvers for building air flow has been carried out. Based on the model used by NEST-Building and the current solver, the properties of the ideal solver have been studied and three non-linear solvers have been proposed. The conclusions of this work can be summarized in the following items:

- The current solver of NEST-Building, based on the SIMPLE method, is really slow because it takes small steps and they have really large computational cost.
- In the model used by NEST-Building for air flow modelling, despite being non-linear, his derivatives can be computed analytically in an easy way because they follow the power rule. The corresponding Jacobian matrix is symmetric and diagonally dominant, this implies that the Jacobian matrix is definite negative.
- The Trust Region Dogleg method has the best of the Steepest Descent method (accuracy) and the Newton method (computational cost). For this reason, is better than the SIMPLE method.
- The Jacobian-free approach is useful when the Jacobian matrix cannot be computed analytically or the number of variables is really large. It is not the case of this work.
- For solving the linear system related to the Newton step, the Conjugate Gradient is fast. The problem is not large enough to need a diagonal preconditioner.
- The Trust Region Dogleg method using the Conjugate Gradient for the Newton step, is accurate and the fastest method studied in this work.

Is left as future work the integration of this new solver to NEST-Building tool. It is proposed to, as in the Fractional Step method used for solving the Navier-Stokes equations, treat the pressure of the rooms as an artificial variable. The pressure should only be used to ensure the mass conservation in the rooms of the building, the pressure actually has no physical meaning. Then, the conservation of energy and momentum should be solved explicitly, as usual, while the mass conservation should be solved implicitly, because the continuity equation is a stiff problem and it needs really small steps to be stable.

9 Bibliography

[1] F. Allard, V. B. Dorer, H. E. Feustel, E. Rodríguez Garcia, M. Grosso, M. K. Herrlin, L. Mingsheng, H. C. Phaff, Y. Utsumi, and H. Yoshino. *Fundamentals of the multizone air flow model-COMIS*. Air Infiltration and Ventilation Centre Coventry, UK, 1990.

[2] H. Boyer, A. P. Lauret, L. Adelard, and T. A. Mara. Building ventilation: a pressure airflow model computer generation and elements of validation. *Energy and Buildings*, 29(3):283–292, 1999.

[3] R. Capdevila, O. Souaihi, J. López, J. Rigola, O. Lehmkuhl, and A. Oliva. Energy simulation of a single family dwelling as a test bench for climate control system assessment. In *REHVA World Congress*, pages 1–10, 2016.

[4] R. Damle. *Modular simulation of thermal systems*. PhD thesis, Universitat Politècnica de Catalunya, Heat and Mass Transfer Technological Center, 2009.

[5] EUROSTAT. Consumption of energy. `http://ec.europa.eu/eurostat/statistics-explained/index.php/Consumption_of_energy#End-users`, 2016. [Accessed 25 November 2016].

[6] J.-M. Fürbringer, C. A. Roulet, and R. Borchiellini. *Evaluation of COMIS. Annex 23: Multizone Air Flow Modelling*. 1996.

[7] D. Kiel and D. Wilson. Gravity driven flows through open doors. In *7th AIVC Conference*, 1986.

[8] D. A. Knoll and D. Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[9] J. Koffi. *Analyse multicritère des stratégies de ventilation en maisons individuelles*. PhD thesis, Université de La Rochelle, 2009.

[10] J. López. *Parallel object-oriented algorithms for simulation of multiphysics. Application to thermal systems*. PhD thesis, Universitat Politècnica de Catalunya, Heat and Mass Transfer Technological Center, 2015.

[11] MathWorks. Equation solving algorithms. `https://es.mathworks.com/help/optim/ug/equation-solving-algorithms.html`. [Accessed 15 December 2016].

[12] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*, volume 113. Springer, 2015.

[13] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[14] OCU. ¿Cuánta energía consume una casa? `https://www.ocu.org/vivienda-y-energia/gas-luz/noticias/cuanta-energia-consume-una-casa-571584`, 2016. [Accessed 7 January 2017].

[15] S. Patankar. *Numerical heat transfer and fluid flow*. CRC press, 1980.

[16] R. H. Perry and D. W. Green. *Perry's chemical engineers' handbook*. McGraw-Hill Professional, 1999.

[17] W. Press, S. Teukolsky, W. Vetterling, and B. Flanner. Numerical recipes in c: The art of scientific computing. 1992.

[18] L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[19] O. Souaihi, J. López, R. Capdevila, J. Rigola, O. Lehmkuhl, and A. Oliva. A complete analysis of the heat, air and moisture transfer on building performance. In *REHVA World Congress*, pages 1–10, 2016.

[20] W. L. Tutorial. Trust region methods. `http://reference.wolfram.com/language/tutorial/UnconstrainedOptimizationTrustRegionMethods.html`. [Accessed 26 December 2016].

[21] W. Ye. Trust region methods. `https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods`, 2014. [Accessed 26 December 2016].